

GBDK 2020 Docs

4.1.0

Generated by Doxygen 1.8.17

Fri Oct 28 2022 00:48:36

1 General Documentation	1
1.1 Introduction	2
1.2 About the Documentation	2
1.3 About GBDK	2
1.4 Historical Info and Links	2
2 Getting Started	2
2.1 1. Download a Release and unzip it	3
2.2 2. Compile Example projects	3
2.2.1 Windows (without Make installed):	3
2.2.2 Linux / MacOS / Windows with Make installed:	3
2.3 3. Use a Template	3
2.4 4. If you use GBTD / GBMB, get the fixed version	3
2.5 5. Review Coding Guidelines	3
2.6 6. Hardware and Resources	4
2.7 7. Set up C Source debugging	4
2.8 8. Try a GBDK Tutorial	4
2.9 9. Read up!	4
2.10 10. Need help?	4
2.11 Migrating From Pre-GBDK-2020 Tutorials	4
2.11.1 Also see:	5
2.11.2 Use auto-banking	5
2.11.3 Non-standard types (UINT8, etc)	5
2.11.4 If using GBTD / GBMB, get the fixed version	5
2.11.5 LCC and SDCC flags that are not needed	5
2.11.6 ROM Header Settings (such as Color, SGB, etc)	5
2.11.7 GBDK Header include changes	5
2.11.8 Include .h headers, not .c source files	5
2.11.9 Use the Template Projects	5
2.11.10 Use hUGTracker instead of gbt_player	6
3 Links and Third-Party Tools	6
3.1 SDCC Compiler Suite User Manual	6
3.2 Getting Help	6
3.3 Game Boy Documentation	6
3.4 Sega Master System / Game Gear Documentation	6
3.5 Tutorials	6
3.6 Example code	7
3.7 Graphics Tools	7
3.8 Music And Sound Effects	7
3.9 Emulators	7
3.10 Debugging tools	8
3.11 Optimizing Assembly	8

3.12 Continuous Integration and Deployment	8
4 Using GBDK	8
4.1 Interrupts	8
4.1.1 Available Interrupts	9
4.1.2 Adding your own interrupt handler	9
4.1.3 Using your own Interrupt Dispatcher	9
4.1.4 Returning from Interrupts and STAT mode	10
4.2 What GBDK does automatically and behind the scenes	10
4.2.1 OAM (VRAM Sprite Attribute Table)	10
4.2.2 Font tiles when using stdio.h	10
4.2.3 Default Interrupt Service Handlers (ISRs)	10
4.2.4 Ensuring Safe Access to Graphics Memory	10
4.3 Copying Functions to RAM and HIRAM	10
4.4 Mixing C and Assembly	11
4.4.1 Inline ASM within C source files	11
4.4.2 In Separate ASM files	11
4.5 Including binary files in C source with incbin	12
4.6 Known Issues and Limitations	12
4.6.1 SDCC	12
5 Coding Guidelines	12
5.1 Learning C / C fundamentals	12
5.1.1 General C tutorials	12
5.1.2 Embedded C introductions	12
5.1.3 Game Boy games in C	12
5.2 Understanding the hardware	12
5.3 Writing optimal C code for the Game Boy and SDCC	12
5.3.1 Tools	12
5.3.2 Variables	13
5.3.3 Code structure	14
5.3.4 GBDK API/Library	14
5.3.5 Toolchain	14
5.3.6 Constants, Signed-ness and Overflows	15
5.3.7 Chars and vararg functions	15
5.4 When C isn't fast enough	16
5.4.1 Calling convention	16
5.4.2 Variables and registers	16
5.4.3 Segments / Areas	16
6 ROM/RAM Banking and MBCs	17
6.1 ROM/RAM Banking and MBCs (Memory Bank Controllers)	17
6.1.1 Non-banked cartridges	17

6.1.2 MBC Banked cartridges (Memory Bank Controllers)	17
6.1.3 Recommended MBC type	17
6.2 Working with Banks	18
6.2.1 Setting the ROM bank for a Source file	18
6.2.2 Setting the RAM bank for a Source file	18
6.2.3 Setting the MBC and number of ROM & RAM banks available	18
6.3 MBC Type Chart	19
6.3.1 Getting Bank Numbers	20
6.3.2 Banking and Functions	20
6.3.3 Const Data (Variables in ROM)	21
6.3.4 Variables in RAM	21
6.3.5 Far Pointers	21
6.3.6 Bank switching	21
6.3.7 Wrapper Function for Accessing Banked Data	21
6.3.8 Currently active bank: <code>_current_bank</code>	21
6.4 Auto-Banking	22
6.5 Errors related to banking (overflow, multiple writes to same location)	22
6.6 Bank space usage	23
6.6.1 Other important notes	23
6.7 Banking example projects	23
7 GBDK Toolchain	23
7.1 Overview	23
7.2 Data Types	23
7.3 Changing Important Addresses	23
7.4 Compiling programs	24
7.4.1 Makefiles	25
7.4.2 Using Makefiles	25
7.4.3 Linker Files and ROM Auto Banking	25
7.5 Build Tools	25
7.5.1 <code>lcc</code>	25
7.5.2 <code>sdcc</code>	25
7.5.3 <code>sdasgb</code>	25
7.5.4 <code>bankpack</code>	26
7.5.5 <code>sldlgb</code>	26
7.5.6 <code>ihxcheck</code>	26
7.5.7 <code>makebin</code>	26
7.6 GBDK Utilities	26
7.6.1 <code>GBCompress</code>	26
7.6.2 <code>png2asset</code>	27
7.6.3 <code>makecom</code>	28
8 Supported Consoles & Cross Compiling	28

8.1 Consoles Supported by GBDK	28
8.2 Cross Compiling for Different Consoles	29
8.2.1 lcc	29
8.2.2 sdcc	29
8.2.3 Console Port and Platform Settings	29
8.3 Cross-Platform Constants	30
8.3.1 Console Identifiers	30
8.3.2 Console Hardware Properties	31
8.4 Using <gbdk/...> headers	31
8.5 Cross Platform Example Projects	31
8.5.1 Cross Platform Asset Example	31
8.6 Porting From Game Boy to Analogue Pocket	31
8.6.1 Registers and Flags	32
8.6.2 Boot logo	32
8.7 Porting From Game Boy to Mega Duck / Cougar Boy	32
8.7.1 Registers and Flags	32
8.8 Porting From Game Boy to SMS/GG	32
8.8.1 Tile Data and Tile Map loading	32
8.9 Hardware Comparison	33
8.9.1 Safe VRAM / Display Controller Access	34
9 Example Programs	34
9.1 banks (various projects)	34
9.2 comm	34
9.3 crash	34
9.4 colorbar	34
9.5 dscan	35
9.6 filltest	35
9.7 fonts	35
9.8 galaxy	35
9.9 gb-dtmf	35
9.10 gbdecompress	35
9.11 irq	35
9.12 large map	35
9.13 metasprites	35
9.14 lcd isr wobble	35
9.15 paint	36
9.16 rand	36
9.17 ram_fn	36
9.18 rpn	36
9.19 samptest	36
9.20 sgb (various)	36

9.21 sound	36
9.22 space	36
9.23 templates	36
10 Frequently Asked Questions (FAQ)	37
10.1 General	37
10.2 Licensing	37
10.3 Graphics and Resources	37
10.4 ROM Header Settings	37
10.5 Errors / Compiling / Toolchain	38
10.6 API / Utilities	39
11 Migrating to new GBDK Versions	40
11.1 GBDK-2020 versions	40
11.1.1 Porting to GBDK-2020 4.1.0	40
11.1.2 Porting to GBDK-2020 4.0.6	40
11.1.3 Porting to GBDK-2020 4.0.5	40
11.1.4 Porting to GBDK-2020 4.0.4	41
11.1.5 Porting to GBDK-2020 4.0.3	41
11.1.6 Porting to GBDK-2020 4.0.2	41
11.1.7 Porting to GBDK-2020 4.0.1	41
11.1.8 Porting to GBDK-2020 4.0	41
11.1.9 Porting to GBDK-2020 3.2	42
11.1.10 Porting to GBDK-2020 3.1.1	42
11.1.11 Porting to GBDK-2020 3.1	42
11.1.12 Porting to GBDK-2020 3.0.1	42
11.2 Historical GBDK versions	42
11.2.1 GBDK 1.1 to GBDK 2.0	42
12 GBDK Release Notes	42
12.1 GBDK-2020 Release Notes	42
12.1.1 GBDK-2020 4.1.0	42
12.1.2 GBDK-2020 4.0.6	44
12.1.3 GBDK-2020 4.0.5	45
12.1.4 GBDK-2020 4.0.4	47
12.1.5 GBDK-2020 4.0.3	48
12.1.6 GBDK-2020 4.0.2	49
12.1.7 GBDK-2020 4.0.1	49
12.1.8 GBDK-2020 4.0	50
12.1.9 GBDK-2020 3.2	50
12.1.10 GBDK-2020 3.1.1	50
12.1.11 GBDK-2020 3.1	51
12.1.12 GBDK-2020 3.0.1	51

12.1.13 GBDK-2020 3.0	51
12.2 Historical GBDK Release Notes	51
12.2.1 GBDK 2.96	51
12.2.2 GBDK 2.95-3	52
12.2.3 GBDK 2.95-2	52
12.2.4 GBDK 2.95	52
12.2.5 GBDK 2.94	53
12.2.6 GBDK 2.93	53
12.2.7 GBDK 2.92-2 for win32	54
12.2.8 GBDK 2.92	54
12.2.9 GBDK 2.91	54
12.2.10 GBDK 2.1.5	55
13 Toolchain settings	55
13.1 lcc settings	55
13.2 sdcc settings	55
13.3 sdasgb settings	57
13.4 sdasz80 settings	58
13.5 bankpack settings	58
13.6 sldlgb settings	58
13.7 sldlz80 settings	59
13.8 ihxcheck settings	59
13.9 makebin settings	59
13.10 makecom settings	60
13.11 gbcompress settings	60
13.12 png2asset settings	60
14 Todo List	61
15 Module Index	61
15.1 C modules	61
16 Data Structure Index	61
16.1 Data Structures	61
17 File Index	61
17.1 File List	61
18 Module Documentation	64
18.1 List of gbdk fonts	64
18.1.1 Description	64
18.1.2 Variable Documentation	64
19 Data Structure Documentation	64
19.1 __far_ptr Union Reference	64

19.1.1 Detailed Description	64
19.1.2 Field Documentation	65
19.2 <code>_fixed</code> Union Reference	65
19.2.1 Detailed Description	65
19.2.2 Field Documentation	65
19.3 <code>atomic_flag</code> Struct Reference	66
19.3.1 Field Documentation	66
19.4 <code>isr_nested_vector_t</code> Struct Reference	66
19.4.1 Field Documentation	66
19.5 <code>isr_vector_t</code> Struct Reference	66
19.5.1 Field Documentation	67
19.6 <code>joypads_t</code> Struct Reference	67
19.6.1 Detailed Description	68
19.6.2 Field Documentation	68
19.7 <code>metasprite_t</code> Struct Reference	68
19.7.1 Detailed Description	69
19.7.2 Field Documentation	70
19.8 <code>OAM_item_t</code> Struct Reference	70
19.8.1 Detailed Description	70
19.8.2 Field Documentation	71
19.9 <code>sfont_handle</code> Struct Reference	71
19.9.1 Detailed Description	71
19.9.2 Field Documentation	71
20 File Documentation	72
20.1 <code>docs/pages/01_getting_started.md</code> File Reference	72
20.2 <code>docs/pages/02_links_and_tools.md</code> File Reference	72
20.3 <code>docs/pages/03_using_gbdk.md</code> File Reference	72
20.4 <code>docs/pages/04_coding_guidelines.md</code> File Reference	72
20.5 <code>docs/pages/05_banking_mbc5.md</code> File Reference	72
20.6 <code>docs/pages/06_toolchain.md</code> File Reference	72
20.7 <code>docs/pages/06b_supported_consoles.md</code> File Reference	72
20.8 <code>docs/pages/07_sample_programs.md</code> File Reference	72
20.9 <code>docs/pages/08_faq.md</code> File Reference	72
20.10 <code>docs/pages/09_migrating_new_versions.md</code> File Reference	72
20.11 <code>docs/pages/10_release_notes.md</code> File Reference	72
20.12 <code>docs/pages/20_toolchain_settings.md</code> File Reference	72
20.13 <code>docs/pages/docs_index.md</code> File Reference	72
20.14 <code>gbdk-lib/include/asm/mos6502/provides.h</code> File Reference	72
20.14.1 Macro Definition Documentation	72
20.15 <code>gbdk-lib/include/asm/sm83/provides.h</code> File Reference	72
20.15.1 Macro Definition Documentation	73

20.16 gbdk-lib/include/asm/z80/provides.h File Reference	73
20.16.1 Macro Definition Documentation	73
20.17 gbdk-lib/include/asm/mos6502/stdarg.h File Reference	73
20.17.1 Macro Definition Documentation	73
20.17.2 Typedef Documentation	74
20.18 gbdk-lib/include/asm/sm83/stdarg.h File Reference	74
20.18.1 Macro Definition Documentation	74
20.18.2 Typedef Documentation	74
20.19 gbdk-lib/include/asm/z80/stdarg.h File Reference	74
20.19.1 Macro Definition Documentation	74
20.19.2 Typedef Documentation	75
20.20 gbdk-lib/include/stdarg.h File Reference	75
20.21 gbdk-lib/include/asm/mos6502/string.h File Reference	75
20.21.1 Detailed Description	75
20.21.2 Function Documentation	75
20.22 gbdk-lib/include/asm/sm83/string.h File Reference	79
20.22.1 Detailed Description	79
20.22.2 Function Documentation	79
20.22.3 Variable Documentation	82
20.23 gbdk-lib/include/asm/z80/string.h File Reference	82
20.23.1 Detailed Description	83
20.23.2 Function Documentation	83
20.24 gbdk-lib/include/string.h File Reference	86
20.24.1 Detailed Description	86
20.25 gbdk-lib/include/asm/mos6502/types.h File Reference	86
20.25.1 Detailed Description	86
20.25.2 Macro Definition Documentation	87
20.25.3 Typedef Documentation	87
20.26 gbdk-lib/include/asm/sm83/types.h File Reference	87
20.26.1 Detailed Description	87
20.26.2 Macro Definition Documentation	88
20.26.3 Typedef Documentation	88
20.27 gbdk-lib/include/asm/types.h File Reference	88
20.27.1 Detailed Description	89
20.27.2 Macro Definition Documentation	89
20.27.3 Typedef Documentation	89
20.28 gbdk-lib/include/asm/z80/types.h File Reference	90
20.28.1 Detailed Description	90
20.28.2 Macro Definition Documentation	90
20.28.3 Typedef Documentation	91
20.29 gbdk-lib/include/types.h File Reference	91
20.29.1 Detailed Description	92

20.29.2 Macro Definition Documentation	92
20.29.3 Typedef Documentation	92
20.30 gbdk-lib/include/assert.h File Reference	92
20.30.1 Macro Definition Documentation	92
20.30.2 Function Documentation	92
20.31 gbdk-lib/include/ctype.h File Reference	92
20.31.1 Detailed Description	93
20.31.2 Function Documentation	93
20.32 gbdk-lib/include/gb/bcd.h File Reference	94
20.32.1 Detailed Description	94
20.32.2 Macro Definition Documentation	95
20.32.3 Typedef Documentation	95
20.32.4 Function Documentation	95
20.33 gbdk-lib/include/gbdk/bcd.h File Reference	96
20.34 gbdk-lib/include/gb/bgb_emu.h File Reference	96
20.34.1 Detailed Description	96
20.35 gbdk-lib/include/gb/cgb.h File Reference	96
20.35.1 Detailed Description	97
20.35.2 Macro Definition Documentation	97
20.35.3 Typedef Documentation	99
20.35.4 Function Documentation	100
20.36 gbdk-lib/include/gb/crash_handler.h File Reference	102
20.36.1 Detailed Description	102
20.36.2 Function Documentation	102
20.37 gbdk-lib/include/gb/drawing.h File Reference	102
20.37.1 Detailed Description	103
20.37.2 Macro Definition Documentation	103
20.37.3 Function Documentation	104
20.38 gbdk-lib/include/gb/emu_debug.h File Reference	107
20.38.1 Detailed Description	107
20.39 gbdk-lib/include/gbdk/emu_debug.h File Reference	107
20.39.1 Detailed Description	107
20.39.2 Macro Definition Documentation	107
20.39.3 Function Documentation	109
20.40 gbdk-lib/include/gb/gb.h File Reference	110
20.40.1 Detailed Description	114
20.40.2 Macro Definition Documentation	114
20.40.3 Typedef Documentation	124
20.40.4 Function Documentation	124
20.40.5 Variable Documentation	151
20.41 gbdk-lib/include/gb/gbdecompress.h File Reference	153
20.41.1 Detailed Description	153

20.41.2 Function Documentation	153
20.41.3 Variable Documentation	156
20.42 gbdk-lib/include/gbdk/gbdecompress.h File Reference	156
20.43 gbdk-lib/include/sms/gbdecompress.h File Reference	156
20.43.1 Function Documentation	156
20.43.2 Variable Documentation	157
20.44 gbdk-lib/include/gb/hardware.h File Reference	157
20.44.1 Detailed Description	162
20.44.2 Macro Definition Documentation	162
20.44.3 Variable Documentation	175
20.45 gbdk-lib/include/msx/hardware.h File Reference	179
20.45.1 Detailed Description	182
20.45.2 Macro Definition Documentation	182
20.45.3 Variable Documentation	188
20.46 gbdk-lib/include/nes/hardware.h File Reference	189
20.46.1 Detailed Description	190
20.46.2 Macro Definition Documentation	190
20.46.3 Variable Documentation	192
20.47 gbdk-lib/include/sms/hardware.h File Reference	192
20.47.1 Detailed Description	195
20.47.2 Macro Definition Documentation	195
20.47.3 Variable Documentation	201
20.48 gbdk-lib/include/gb/isr.h File Reference	202
20.48.1 Detailed Description	202
20.48.2 Macro Definition Documentation	202
20.48.3 Typedef Documentation	203
20.49 gbdk-lib/include/gb/metasprites.h File Reference	203
20.49.1 Detailed Description	204
20.49.2 Metasprite support	204
20.49.3 Metasprites composed of variable numbers of sprites	204
20.49.4 Metasprites and sprite properties (including cgb palette)	205
20.49.5 Macro Definition Documentation	205
20.49.6 Typedef Documentation	205
20.49.7 Function Documentation	206
20.49.8 Variable Documentation	208
20.50 gbdk-lib/include/gbdk/metasprites.h File Reference	209
20.51 gbdk-lib/include/msx/metasprites.h File Reference	209
20.51.1 Macro Definition Documentation	209
20.51.2 Typedef Documentation	210
20.51.3 Function Documentation	210
20.51.4 Variable Documentation	211
20.52 gbdk-lib/include/nes/metasprites.h File Reference	211

20.52.1 Detailed Description	212
20.52.2 Metasprite support	212
20.52.3 Metasprites composed of variable numbers of sprites	212
20.52.4 Metasprites and sprite properties (including cgb palette)	213
20.52.5 Macro Definition Documentation	213
20.52.6 Typedef Documentation	213
20.52.7 Function Documentation	213
20.52.8 Variable Documentation	216
20.53 gbdk-lib/include/sms/metasprites.h File Reference	217
20.53.1 Detailed Description	217
20.53.2 Metasprite support	217
20.53.3 Metasprites composed of variable numbers of sprites	217
20.53.4 Metasprite support	218
20.53.5 Metasprites composed of variable numbers of sprites	218
20.53.6 Macro Definition Documentation	218
20.53.7 Typedef Documentation	218
20.53.8 Function Documentation	218
20.53.9 Variable Documentation	219
20.54 gbdk-lib/include/gb/sgb.h File Reference	220
20.54.1 Detailed Description	221
20.54.2 Macro Definition Documentation	221
20.54.3 Function Documentation	222
20.54.4 Variable Documentation	223
20.55 gbdk-lib/include/gbdk/console.h File Reference	223
20.55.1 Detailed Description	223
20.55.2 Function Documentation	223
20.56 gbdk-lib/include/gbdk/far_ptr.h File Reference	224
20.56.1 Detailed Description	224
20.56.2 Macro Definition Documentation	224
20.56.3 Typedef Documentation	226
20.56.4 Function Documentation	226
20.56.5 Variable Documentation	226
20.57 gbdk-lib/include/gbdk/font.h File Reference	227
20.57.1 Detailed Description	227
20.57.2 Macro Definition Documentation	227
20.57.3 Typedef Documentation	228
20.57.4 Function Documentation	228
20.58 gbdk-lib/include/gbdk/gbdk-lib.h File Reference	229
20.58.1 Detailed Description	229
20.59 gbdk-lib/include/gbdk/incbin.h File Reference	229
20.59.1 Detailed Description	229
20.59.2 Macro Definition Documentation	229

20.60 gbdk-lib/include/gbdk/platform.h File Reference	230
20.61 gbdk-lib/include/gbdk/rledecompress.h File Reference	230
20.61.1 Detailed Description	231
20.61.2 Macro Definition Documentation	231
20.61.3 Function Documentation	231
20.62 gbdk-lib/include/gbdk/version.h File Reference	231
20.62.1 Macro Definition Documentation	232
20.63 gbdk-lib/include/limits.h File Reference	232
20.63.1 Macro Definition Documentation	232
20.64 gbdk-lib/include/msx/msx.h File Reference	233
20.64.1 Detailed Description	236
20.64.2 Macro Definition Documentation	237
20.64.3 Typedef Documentation	243
20.64.4 Function Documentation	243
20.64.5 Variable Documentation	254
20.65 gbdk-lib/include/nes/nes.h File Reference	256
20.65.1 Detailed Description	259
20.65.2 Macro Definition Documentation	259
20.65.3 Typedef Documentation	266
20.65.4 Function Documentation	267
20.65.5 Variable Documentation	284
20.66 gbdk-lib/include/nes/rgb_to_nes_macro.h File Reference	284
20.66.1 Macro Definition Documentation	284
20.67 gbdk-lib/include/rand.h File Reference	284
20.67.1 Detailed Description	285
20.67.2 Macro Definition Documentation	285
20.67.3 Function Documentation	285
20.67.4 Variable Documentation	286
20.68 gbdk-lib/include/setjmp.h File Reference	286
20.68.1 Macro Definition Documentation	286
20.68.2 Typedef Documentation	287
20.68.3 Function Documentation	287
20.69 gbdk-lib/include/sms/sms.h File Reference	287
20.69.1 Detailed Description	291
20.69.2 Macro Definition Documentation	291
20.69.3 Typedef Documentation	297
20.69.4 Function Documentation	298
20.69.5 Variable Documentation	309
20.70 gbdk-lib/include/stdatomic.h File Reference	310
20.70.1 Function Documentation	310
20.71 gbdk-lib/include/stdbool.h File Reference	311
20.71.1 Macro Definition Documentation	311

20.72 gbdk-lib/include/stddef.h File Reference	311
20.72.1 Macro Definition Documentation	311
20.72.2 Typedef Documentation	312
20.73 gbdk-lib/include/stdint.h File Reference	312
20.73.1 Macro Definition Documentation	313
20.73.2 Typedef Documentation	316
20.74 gbdk-lib/include/stdio.h File Reference	318
20.74.1 Detailed Description	318
20.74.2 Function Documentation	318
20.75 gbdk-lib/include/stdlib.h File Reference	319
20.75.1 Macro Definition Documentation	320
20.75.2 Function Documentation	320
20.76 gbdk-lib/include/stdnoreturn.h File Reference	323
20.76.1 Macro Definition Documentation	323
20.77 gbdk-lib/include/time.h File Reference	323
20.77.1 Detailed Description	324
20.77.2 Macro Definition Documentation	324
20.77.3 Typedef Documentation	324
20.77.4 Function Documentation	324
20.78 gbdk-lib/include/typeof.h File Reference	324
20.78.1 Macro Definition Documentation	325
Index	327

1 General Documentation

- [Getting Started](#)
- [Links and Third-Party Tools](#)
- [Using GBDK](#)
- [Coding Guidelines](#)
- [ROM/RAM Banking and MBCs](#)
- [Supported Consoles & Cross Compiling](#)
- [GBDK Toolchain](#)
- [Example Programs](#)
- [Frequently Asked Questions \(FAQ\)](#)
- [Migrating to new GBDK Versions](#)
- [GBDK Release Notes](#)
- [Toolchain settings](#)

1.1 Introduction

Welcome to GBDK-2020! The best thing to do is head over to the [Getting Started](#) section to get up and running.

If you are upgrading please check [GBDK Release Notes](#) and [Migrating to new GBDK Versions](#)

1.2 About the Documentation

This documentation is partially based on material written by the original GBDK authors in 1999 and updated for GBDK-2020. The API docs are automatically generated from the C header files using Doxygen.

GBDK-2020 is an updated version of the original GBDK with a modernized SDCC toolchain and many API improvements and fixes. It can be found at: <https://github.com/gbdk-2020/gbdk-2020/>.

The original GBDK sources, documentation and website are at: <http://gbdk.sourceforge.net/>

1.3 About GBDK

The GameBoy Developer's Kit (GBDK, GBDK-2020) is used to develop games and programs for the Nintendo Game Boy (and some other consoles) in C and assembly. GBDK includes a set of libraries for the most common requirements and generates image files for use with a real GameBoy or emulators.

GBDK features:

- C and ASM toolchain based on SDCC with some support utilities
- A set of libraries with source code
- Example programs in ASM and in C
- Support for multiple ROM bank images and auto-banking
- Support for multiple consoles: Game Boy, Analogue Pocket, Mega Duck, Master System and Game Gear

GBDK is freeware. Most of the tooling code is under the GPL. The runtime libraries should be under the LGPL. Please consider mentioning GBDK in the credits of projects made with it.

1.4 Historical Info and Links

Work on the original GBDK (pre-2020) was by:

Pascal Felber, Lars Malmberg, Michael Hope, David Galloway (djmips), John Fuge, and others.

The following is from the original GBDK documentation:

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.

quangDX.com

[The \(original\) gbdk homepage](#)

[Jeff Frohwein's GB development page](#). A extensive source of Game Boy related information, including GeeBee's GB faq and the pan/k00Pa document.

2 Getting Started

Follow the steps in this section to start using GBDK-2020.

2.1 1. Download a Release and unzip it

You can get the latest releases from here: <https://github.com/gbdk-2020/gbdk-2020/releases>

2.2 2. Compile Example projects

Make sure your GBDK-2020 installation is working correctly by compiling some of the included [example projects](#). If everything works in the steps below and there are no errors reported then each project that was built should have its own .gb ROM file (or suitable extension for the other supported targets).

2.2.1 Windows (without Make installed):

Navigate to a project within the example projects folder ("examples\gb\" under your GBDK-2020 install folder) and open a command line. Then type:

```
compile
```

or

```
compile.bat
```

This should build the example project. You can also navigate into other example project folders and build in the same way.

2.2.2 Linux / MacOS / Windows with Make installed:

Navigate to the example projects folder ("examples/gb/" under your GBDK-2020 install folder) and open a command line. Then type:

```
make
```

This should build all of the examples sequentially. You can also navigate into an individual example project's folder and build it by typing `make`.

2.3 3. Use a Template

To create a new project use a template!

There are template projects included in the [GBDK example projects](#) to help you get up and running. Their folder names start with `template_`.

1. Copy one of the template folders to a new folder name.
2. If you moved the folder out of the GBDK examples then you **must** update the GBDK path variable and/or the path to LCC in the Makefile or `compile.bat` so that it will still build correctly.
3. Type `make` on the command line in that folder to verify it still builds.
4. Open `main.c` to start making changes.

2.4 4. If you use GBTD / GBMB, get the fixed version

If you plan to use GBTD / GBMB for making graphics, make sure to get the version with the `const` fix and other improvements. See [const_gbtd_gbmb](#).

2.5 5. Review Coding Guidelines

Take a look at the [coding guidelines](#), even if you have experience writing software for other platforms. There is important information to help you get good results and performance on the Game Boy.

If you haven't written programs in C before, check the [C tutorials section](#).

2.6 6. Hardware and Resources

If you have a specific project in mind, consider what hardware you want to target. It isn't something that has to be decided up front, but it can influence design and implementation.

What size will your game or program be?

- 32K Cart (no-MBC required)
- Larger than 32K (MBC required)
- See more details about [ROM Banking and MBCs](#)

What console platform(s) will it run on?

- Game Boy (GB/GBC)
- Analogue Pocket (AP)
- Sega Master System (SMS)
- Game Gear (GG)
- Mega Duck (DUCK)
- See [Supported Consoles & Cross Compiling](#)

If targeting the Game Boy, what hardware will it run on?

- Game Boy (& Game Boy Color)
- Game Boy Color only
- Game Boy & Super Game Boy
- See how to [set the compatibility type in the cartridge header](#). Read more about hardware differences in the [Pandocs](#)

2.7 7. Set up C Source debugging

Tracking down problems in code is easier with a debugger. Emulicious has a [debug adapter](#) that provides C source debugging with GBDK-2020.

2.8 8. Try a GBDK Tutorial

You might want to start off with a guided GBDK tutorial from the [GBDK Tutorials section](#).

- **Note:** Tutorials (or parts of them) may be based on the older GBDK from the 2000's before it was updated to be GBDK-2020. The general principles are all the same, but the setup and parts of the [toolchain](#) (compiler/etc) may be somewhat different and some links may be outdated (pointing to the old GBDK or old tools).

2.9 9. Read up!

- It is strongly encouraged to read more [GBDK-2020 General Documentation](#).
- Learn about the Game Boy hardware by reading through the [Pandocs](#) technical reference.

2.10 10. Need help?

Check out the links for [online community and support](#) and read the [FAQ](#).

2.11 Migrating From Pre-GBDK-2020 Tutorials

Several popular GBDK Tutorials, Videos and How-to's were made before GBDK-2020 was available, as a result some information they include is outdated or incompatible. The following summarizes changes that should be made for best results.

2.11.1 Also see:

- [Migrating to new GBDK Versions](#)
- [Coding Guidelines](#)
- [Getting Started](#) (the section above this)

2.11.2 Use auto-banking

GBDK-2020 now supports auto-banking ([rom_autobanking](#)). In most cases using auto-banking will be easier and less error prone than manually assigning source and assets to banks.

- There is a source example `banks_autobank` project.

2.11.3 Non-standard types (UINT8, etc)

The old GBDK types `UINT8`, `INT8`, `UINT16`, `INT16` are non-standard and less portable.

The following should be used instead: `uint8_t`, `int16_t`, `uint16_t`, `int32_t`, `uint32_t` and `bool`.

These are standard types defined in `stdint.h` (`#include <stdint.h>`) and `stdbool.h` (`#include <stdbool.h>`).

2.11.4 If using GBTD / GBMB, get the fixed version

If you plan to use GBTD / GBMB for making graphics, make sure to get the version with the `const` fix and other improvements. See [const_gbtd_gbmb](#).

2.11.5 LCC and SDCC flags that are not needed

The following flag is no longer needed with `lcc` and `sdcc`, it can be removed without any loss of performance.

- `-DUSE_SFR`
 - Behavior formerly enabled by `USE_SFR_FOR_REG` is on by default now (no need to specify it, it isn't a tested `#ifdef` anymore). Check here why: <https://gbdev.gg8.se/forums/viewtopic.php?id=697>

2.11.6 ROM Header Settings (such as Color, SGB, etc)

Setting ROM bytes directly with `-Wl-yp0x<address>=0x<value>` is no longer supported. Instead use `makebin` flags. For example, use `-Wm-yC` instead of `-Wl-yp0x143=0xC0`. See [faq_gb_type_header_setting](#).

2.11.7 GBDK Header include changes

The following header files which are now cross platform were moved from `gb/` to `gbdk/`: `bcd.h`, `console.h`, `far_ptr.h`, `font.h`, `gbdecompress.h`, `gbdk-lib.h`, `incbin.h`, `metasprites.h`, `platform.h`, `version.h`

- When including them use `#include <gbdk/...>` instead of `#include <gb/>`

2.11.8 Include .h headers, not .c source files

Do not `#include` `.c` source files into other `.c` source files. Instead create `.h` header files for them and include those.

- https://www.tutorialspoint.com/cprogramming/c_header_files.htm

2.11.9 Use the Template Projects

Modern project templates are included with GBDK-2020. Using them (and their `Makefile` or `compile.bat`) as a starting point for projects is recommended and can help ensure better default settings and project organization.

2.11.10 Use hUGTracker instead of gbt_player

hUGTracker and its driver [hUGDriver](#) are smaller, more efficient and more versatile than gbt_player.

3 Links and Third-Party Tools

This is a brief list of useful tools and information. It is not meant to be complete or exhaustive, for a larger list see the [Awesome Game Boy Development](#) list.

3.1 SDCC Compiler Suite User Manual

- GBDK-2020 uses the SDCC compiler and related tools. The SDCC manual goes into much more detail about available features and how to use them.
<http://sdcc.sourceforge.net/doc/sdccman.pdf>
<http://sdcc.sourceforge.net>
- The SDCC assembler and linker (sdas / asxxxx and aslink) manual.
<https://sourceforge.net/p/sdcc/code/HEAD/tree/trunk/sdcc/sdas/doc/asmlnk.txt>

3.2 Getting Help

- GBDK Discord community:
<https://github.com/gbdk-2020/gbdk-2020/#discord-servers>
- Game Boy discussion forum:
<https://gbdev.gg8.se/forums/>

3.3 Game Boy Documentation

- **Pandocs**
Extensive and up-to-date technical documentation about the Game Boy and related hardware.
<https://gbdev.io/pandocs/>
- **Awesome Game Boy Development list**
A list of Game Boy/Color development resources, tools, docs, related projects and homebrew.
<https://gbdev.io/resources.html>

3.4 Sega Master System / Game Gear Documentation

- **SMS Power!**
Community site with technical documentation, reviews and other content related to the Sega 8-bit systems.
<https://www.smspower.org/>

3.5 Tutorials

- **Larold's Jubilant Junkyard Tutorials**
Several walk throughs about the fundamentals of developing for the Game Boy with GBDK-2020. There are simple examples with source code.
<https://laroldsjubilantjunkyard.com/tutorials/>
- **Gaming Monsters Tutorials**
Several video tutorials and code for making games with GBDK/GBDK-2020.
<https://www.youtube.com/playlist?list=PLeEj4c2zF7PaFv5MPYhNAkBGGrkx4iPGJo>
<https://github.com/gingemonster/GamingMonstersGameBoySampleCode>

- **Pocket Leage Tutorial**

<https://blog.ty-porter.dev/development/2021/04/04/writing-a-gameboy-game-in-2021-pt-1.html>

3.6 Example code

- **Simplified GBDK examples**

https://github.com/mrombout/gbdk_playground/commits/master

3.7 Graphics Tools

- **Game Boy Tile Designer and Map Builder (GBTD / GBMB)**

Sprite / Tile editor and Map Builder that can export to C that works with GBDK.

This is an updated version with const export fixed and other improvements.

https://github.com/gbdk-2020/GBTD_GBMB

- A GIMP plugin to read/write GBR/GBM files and do map conversion:

<https://github.com/bbbbr/gimp-tilemap-gb>

- Command line version of the above tool that doesn't require GIMP (png2gbtiles):

<https://github.com/bbbbr/gimp-tilemap-gb/tree/master/console>

- **Tilemap Studio**

A tilemap editor for Game Boy, GBC, GBA, or SNES projects.

<https://github.com/Rangi42/tilemap-studio/>

3.8 Music And Sound Effects

- **hUGETRacker and hUGEdriver**

A tracker and music driver that work with GBDK and RGBDS. It is smaller, more efficient and more versatile than gbt_player.

<https://github.com/untoxa/hUGEBuild>

<https://github.com/SuperDisk/hUGEDriver>

<https://github.com/SuperDisk/hUGETRacker>

- **CBT-FX**

A sound effects driver which can play effects created in FX Hammer. <https://github.com/datguywitha3ds/CBT-FX>

- **VGM2GBSFX**

A sound effects converter and driver for DMG VGM files, FX Hammer and PCM WAV files. <https://github.com/untoxa/VGM2GBSFX>

- **GBT Player**

A .mod converter and music driver that works with GBDK and RGBDS.

<https://github.com/AntonioND/gbt-player>

Docs from GBStudio that should mostly apply: <https://www.gbstudio.dev/docs/music/>

3.9 Emulators

- **BGB**

Accurate emulator, has useful debugging tools.

<http://bgb.bircd.org/>

- **Emulicious**

An accurate emulator with extensive tools including source level debugging.

<https://emulicious.net/>

3.10 Debugging tools

- **Emulicious debug adapter**

Provides source-level debugging in VS Code that works with GBDK2020.

<https://marketplace.visualstudio.com/items?itemName=emulicious.emulicious-debugger>

- **romusage**

Calculate used and free space in banks (ROM/RAM) and warn about errors such as bank overflows.

<https://github.com/bbbbr/romusage>

- **.noi file to sym conversion for bgb**

Debug information in .noi files can be converted to a symbol format that BGB recognizes using:

- `lcc` : `-Wm-yS` (with `--debug`, or `-Wl-j` to create the .noi)
- directly with `makebin` : `-yS` (with `-j` passed to the linker)

- **src2sym.pl**

Add line-by-line C source code to the main symbol file in a BGB compatible format. This allows for C source-like debugging in BGB in a limited way.

<https://gbdev.gg8.se/forums/viewtopic.php?id=710>

3.11 Optimizing Assembly

- **Optimizing Assembly Code**

Pret has a useful guide to optimizing assembly for the Game Boy for times when asm using in a project in addition to C. <https://github.com/pret/pokecrystal/wiki/Optimizing-assembly-code>

3.12 Continuous Integration and Deployment

- **GBDK GitHub Action Builder**

A Github Action which provides basic CI/CD for building projects based on GBDK (not for building GBDK itself).

<https://github.com/wujoood/gbdk-2020-github-builder>

4 Using GBDK

4.1 Interrupts

Interrupts allow execution to jump to a different part of your code as soon as an external event occurs - for example the LCD entering the vertical blank period, serial data arriving or the timer reaching its end count. For an example see the `irq.c` sample project.

Interrupts in GBDK are handled using the functions `disable_interrupts()`, `enable_interrupts()`, `set_interrupts(uint8_t ier)` and the interrupt service routine (ISR) linkers `add_VBL()`, `add_TIM`, `add_low_priority_TIM`, `add_LCD`, `add_SIO` and `add_JOY` which add interrupt handlers for the vertical blank, timer, LCD, serial link and joypad interrupts respectively.

Since an interrupt can occur at any time an Interrupt Service Request (ISR) cannot take any arguments or return anything. Its only way of communicating with the greater program is through the global variables. When interacting with those shared ISR global variables from main code outside the interrupt, it is a good idea to wrap them in a `critical {}` section in case the interrupt occurs and modifies the variable while it is being used.

Interrupts should be disabled before adding ISRs. To use multiple interrupts, *logical OR* the relevant IFLAGS together.

ISRs should be kept as small and short as possible, do not write an ISR so long that the Game Boy hardware spends all of its time servicing interrupts and has no time spare for the main code.

For more detail on the Game Boy interrupts consider reading about them in the [Pandocs](#).

4.1.1 Available Interrupts

The GameBoy hardware can generate 5 types of interrupts. Custom Interrupt Service Routines (ISRs) can be added in addition to the built-in ones available in GBDK.

- VBL : LCD Vertical Blanking period start
 - The default VBL ISR is installed automatically.
 - * See [add_VBL\(\)](#) and [remove_VBL\(\)](#)
- LCD : LCDC status (such as the start of a horizontal line)
 - See [add_LCD\(\)](#) and [remove_LCD\(\)](#)
 - Example project: `lcd_isr_wobble`
- TIM : Timer overflow
 - See [add_TIM\(\)](#) (or [add_low_priority_TIM\(\)](#)) and [remove_TIM\(\)](#)
 - Example project: `tim`
- SIO : Serial Link I/O transfer end
 - The default SIO ISR gets installed automatically if any of the standard SIO calls are used. These calls include [add_SIO\(\)](#), [remove_SIO\(\)](#), [send_byte\(\)](#), [receive_byte\(\)](#).
 - The default SIO ISR cannot be removed once installed. Only secondary chained SIO ISRs (added with [add_SIO\(\)](#)) can be removed.
 - See [add_SIO\(\)](#) and [remove_SIO\(\)](#)
 - Example project: `comm`
- JOY : Transition from high to low of a joypad button
 - See [add_JOY\(\)](#) and [remove_JOY\(\)](#)

4.1.2 Adding your own interrupt handler

It is possible to install your own interrupt handlers (in C or in assembly) for any of these interrupts. Up to 4 chained 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 for VBL.

Interrupt handlers are called in sequence. To install a new interrupt handler, do the following:

1. Write a function (say `foo()`) that takes no parameters, and that returns nothing. Remember that the code executed in an interrupt handler must be short.
2. Inside a `__critical { ... }` section, install your interrupt handling routines using the `add_XXX()` function, where XXX is the interrupt that you want to handle.
3. Enable interrupts for the IRQ you want to handle, using the [set_interrupts\(\)](#) function. Note that the VBL interrupt is already enabled before the `main()` function is called. If you want to set the interrupts before `main()` is called, you must install an initialization routine.

See the `irq` example project for additional details for a complete example.

4.1.3 Using your own Interrupt Dispatcher

If you want to use your own Interrupt Dispatcher instead of the GBDK chained dispatcher (for improved performance), then don't call the `add_...()` function for the respective interrupt and its dispatcher won't be installed.

- Exception: the VBL dispatcher will always be linked in at compile time.
- For the SIO interrupt, also do not make any standard SIO calls to avoid having its dispatcher installed.

Then, [ISR_VECTOR\(\)](#) or [ISR_NESTED_VECTOR\(\)](#) can be used to install a custom ISR handler.

4.1.4 Returning from Interrupts and STAT mode

By default when an Interrupt handler completes and is ready to exit it will check `STAT_REG` and only return at the BEGINNING of either LCD Mode 0 or Mode 1. This helps prevent graphical glitches caused when an ISR interrupts a graphics operation in one mode but returns in a different mode for which that graphics operation is not allowed. You can change this behavior using `nowait_int_handler()` which does not check `STAT_REG` before returning. Also see `wait_int_handler()`.

4.2 What GBDK does automatically and behind the scenes

4.2.1 OAM (VRAM Sprite Attribute Table)

GBDK sets up a Shadow OAM which gets copied automatically to the hardware OAM by the default V-Blank ISR. The Shadow OAM allows updating sprites without worrying about whether it is safe to write to them or not based on the hardware LCD mode.

4.2.2 Font tiles when using `stdio.h`

Including `stdio.h` and using functions such as `printf()` will use a large number of the background tiles for font characters. If `stdio.h` is not included then that space will be available for use with other tiles instead.

4.2.3 Default Interrupt Service Handlers (ISRs)

- V-Blank: A default V-Blank ISR is installed on startup which copies the Shadow OAM to the hardware OAM and increments the global `sys_time` variable once per frame.
- Serial Link I/O: If any of the GBDK serial link functions are used such as `send_byte()` and `receive_byte()`, the default SIO serial link handler will be installed automatically at compile-time.
- APA Graphics Mode: When this mode is used (via `drawing.h`) custom VBL and LCD ISRs handlers will be installed (`drawing_vbl` and `drawing_lcd`). Changing the mode to `(mode(M_TEXT_OUT) ;)` will cause them to be de-installed. These handlers are used to change the tile data source at start-of-frame and mid-frame so that 384 background tiles can be used instead of the typical 256.

4.2.4 Ensuring Safe Access to Graphics Memory

There are certain times during each video frame when memory and registers relating to graphics are "busy" and should not be read or written to (otherwise there may be corrupt or dropped data). GBDK handles this automatically for most graphics related API calls. It also ensures that ISR handlers return in such a way that if they interrupted a graphics access then it will only resume when access is allowed.

The ISR return behavior [can be turned off](#) using the `nowait_int_handler`.

For more details see the related Pandocs section: https://gbdev.io/pandocs/Accessing_VRAM_and_OAM.html

4.3 Copying Functions to RAM and HIRAM

See the `ram_function` example project included with GBDK which demonstrates copying functions to RAM and HIRAM.

Warning! Copying of functions is generally not safe since they may contain jumps to absolute addresses that will not be converted to match the new location.

It is possible to copy functions to RAM and HIRAM (using the `memcpy()` and `hramcpy()` functions), and execute them from C. Ensure you have enough free space in RAM or HIRAM for copying a function.

There are basically two ways for calling a function located in RAM, HIRAM, or ROM:

- Declare a pointer-to-function variable, and set it to the address of the function to call.
- Declare the function as extern, and set its address at link time using the `-Wl-gXXX=#` flag (where XXX is the name of the function, and # is its address).

The second approach is slightly more efficient. Both approaches are demonstrated in the `ram_function.c` example.

4.4 Mixing C and Assembly

You can mix C and assembly (ASM) in two ways as described below. For additional detail see the [links_sdcc_docs](#).

4.4.1 Inline ASM within C source files

Example:

```
__asm__("nop");
```

Another Example:

```
void some_c_function()
{
    // Optionally do something
    __asm
        (ASM code goes here)
    __endasm;
}
```

4.4.2 In Separate ASM files

Todo This is from GBDK 2.x docs, verify it with GBDK-2020 and modern SDCC

It is possible to assemble and link files written in ASM alongside files written in C.

- A C identifier `i` will be called `_i` in assembly.
- Results are always returned into the `DE` register.
- Parameters are passed on the stack (starting at `SP+2` because the return address is also saved on the stack).
- Assembly identifiers are exported using the `.globl` directive.
- You can access GameBoy hardware registers using `_reg_0xXX` where `XX` is the register number (see `sound.c` for an example).
- Registers must be preserved across function calls (you must store them at function begin, and restore them at the end), except `HL` (and `DE` when the function returns a result).

Here is an example of how to mix assembly with C:

`main.c`

```
main()
{
    int16_t i;
    int16_t add(int16_t, int16_t);

    i = add(1, 3);
}
```

`add.s`

```
.globl _add
_add:                ; int16_t add(int16_t a, int16_t b)
                    ; There is no register to save:
                    ; BC is not used
                    ; DE is the return register
                    ; HL needs never to be saved

LDA    HL, 2(SP)
LD     E, (HL)      ; Get a in DE
INC    HL
LD     D, (HL)
INC    HL
LD     A, (HL)      ; Get b in HL
INC    HL
LD     H, (HL)
LD     L, A
ADD    HL, DE       ; Add DE to HL
LD     D, H
LD     E, L

                    ; There is no register to restore
RET                                ; Return result in DE
```


4.5 Including binary files in C source with incbin

Data from binary files can be included in C source files as a const array using the `INCBIN()` macro. See the `incbin` example project for a demo of how to use it.

4.6 Known Issues and Limitations

4.6.1 SDCC

- Const arrays declared with `somevar[n] = {x}` will **NOT** get initialized with value `x`. This may change when the SDCC RLE initializer is fixed. Use `memset` for now if you need it.
- SDCC banked calls and [far pointers](#) in GBDK only save one byte for the ROM bank, so for example they are limited to **bank 15** max for MBC1 and **bank 255** max for MBC5. See [banked_calls](#) for more details.

5 Coding Guidelines

5.1 Learning C / C fundamentals

Writing games and other programs with GBDK will be much easier with a basic understanding of the C language. In particular, understanding how to use C on "Embedded Platforms" (small computing systems, such as the Game Boy) can help you write better code (smaller, faster, less error prone) and avoid common pitfalls.

5.1.1 General C tutorials

- <https://www.learn-c.org/>
- <https://www.tutorialspoint.com/cprogramming/index.htm>
- <https://www.chiark.greenend.org.uk/~sgtatham/cdescent/>

5.1.2 Embedded C introductions

- <http://dsp-book.narod.ru/CPES.pdf>
- <https://www.phaedsys.com/principals/bytecraft/bytecraftdata/bcfirststeps.pdf>

5.1.3 Game Boy games in C

- <https://gbdev.io/resources.html#c>

5.2 Understanding the hardware

In addition to understanding the C language it's important to learn how the Game Boy hardware works. What it is capable of doing, what it isn't able to do, and what resources are available to work with. A good way to do this is by reading the [Pandocs](#) and checking out the [awesome_gb](#) list.

5.3 Writing optimal C code for the Game Boy and SDCC

The following guidelines can result in better code for the Game Boy, even though some of the guidance may be contrary to typical advice for general purpose computers that have more resources and speed.

5.3.1 Tools

5.3.1.1 GBTD / GBMB, Arrays and the "const" keyword **Important:** The old [GBTD/GBMB](#) fails to include the `const` keyword when exporting to C source files for GBDK. That causes arrays to be created in RAM instead of ROM, which wastes RAM, uses a lot of ROM to initialize the RAM arrays and slows the compiler down a lot.

___Use of [toxa's updated GBTD/GBMB](#) is highly recommended.___

If you wish to use the original tools, you must add the `const` keyword every time the graphics are re-exported to C source files.

5.3.2 Variables

- Use 8-bit values as much as possible. They will be much more efficient and compact than 16 and 32 bit types.
- Prefer unsigned variables to signed ones: the code generated will be generally more efficient, especially when comparing two values.
- Use explicit types so you always know the size of your variables. `int8_t`, `uint8_t`, `int16_t`, `uint16_t`, `int32_t`, `uint32_t` and `bool`. These are standard types defined in `stdint.h` (`#include <stdint.h>`) and `stdbool.h` (`#include <stdbool.h>`).
- Global and local static variables are generally more efficient than local non-static variables (which go on the stack and are slower and can result in slower code).
- `const` keyword: use `const` for arrays, structs and variables with read-only (constant) data. It will reduce ROM, RAM and CPU usage significantly. Non-`const` values are loaded from ROM into RAM inefficiently, and there is no benefit in loading them into the limited available RAM if they aren't going to be changed.
- Here is how to declare `const` pointers and variables:
 - non-const pointer to a const variable: `const uint8_t * some_pointer;`
 - const pointer to a non-const variable: `uint8_t * const some_pointer;`
 - const pointer to a const variable: `const uint8_t * const some_pointer;`
 - <https://codeforwin.org/2017/11/constant-pointer-and-pointer-to-constant-in-c.html>
 - <https://stackoverflow.com/questions/21476869/constant-pointer-vs-pointer-to-const>
- For calculated values that don't change, pre-compute results once and store the result. Using lookup-tables and the like can improve speed and reduce code size. Macros can sometimes help. It may be beneficial to do the calculations with an outside tool and then include the result as C code in a `const` array.
- Use an advancing pointer (`someStruct->var = x; someStruct++`) to loop through arrays of structs instead of using indexing each time in the loop `someStruct[i].var = x`.
- When modifying variables that are also changed in an Interrupt Service Routine (ISR), wrap them the relevant code block in a `__critical { }` block. See <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.3.9>
- When using constants and literals the `U`, `L` and `UL` postfixes can be used.
 - `U` specifies that the constant is unsigned
 - `L` specifies that the constant is long.
 - NOTE: In SDCC 3.6.0, the default for `char` changed from signed to unsigned. The manual says to use `--fsigned-char` for the old behavior, this option flag is included by default when compiling through `lcc`.
- A fixed point type (`fixed`) is included with GBDK when precision greater than whole numbers is required for 8 bit range values (since floating point is not included in GBDK).

See the "Simple Physics" sub-pixel example project.
Code example:

```
fixed player[2];
...
// Modify player position using its 16 bit representation
player[0].w += player_speed_x;
player[1].w += player_speed_y;
...
// Use only the upper 8 bits for setting the sprite position
move_sprite(0, player[0].h, player[1].h);
```

5.3.3 Code structure

- Do not `#include .c` source files into other `.c` source files. Instead create `.h` header files for them and include those. https://www.tutorialspoint.com/cprogramming/c_header_files.htm
- Instead of using a blocking `delay()` for things such as sprite animations/etc (which can prevent the rest of the game from continuing) many times it's better to use a counter which performs an action once every N frames. `sys_time` may be useful in these cases.
- When processing for a given frame is done and it is time to wait before starting the next frame, `wait_vbl_done()` can be used. It uses HALT to put the CPU into a low power state until processing resumes. The CPU will wake up and resume processing at the end of the current frame when the Vertical Blanking interrupt is triggered.
- Minimize use of multiplication, modulo with non-powers of 2, and division with non-powers of 2. These operations have no corresponding CPU instructions (software functions), and hence are time costly.
 - SDCC has some optimizations for:
 - * Division by powers of 2. For example `n /= 4u` will be optimized to `n >>= 2`.
 - * Modulo by powers of 2. For example: `(n % 8)` will be optimized to `(n & 0x7)`.
 - If you need decimal numbers to count or display a score, you can use the GBDK BCD (`binary coded decimal`) number functions. See: `bcd.h` and the BCD example project included with GBDK.
- Avoid long lists of function parameters. Passing many parameters can add overhead, especially if the function is called often. Globals and local static vars can be used instead when applicable.
- Use inline functions if the function is short (with the `inline` keyword, such as `inline uint8_t myFunction() { ... }`).
- Do not use recursive functions.

5.3.4 GBDK API/Library

- `stdio.h`: If you have other ways of printing text, avoid including `stdio.h` and using functions such as `printf()`. Including it will use a large number of the background tiles for font characters. If `stdio.h` is not included then that space will be available for use with other tiles instead.
- `drawing.h`: The Game Boy graphics hardware is not well suited to frame-buffer style graphics such as the kind provided in `drawing.h`. Due to that, most drawing functions (rectangles, circles, etc) will be slow. When possible it's much faster and more efficient to work with the tiles and tile maps that the Game Boy hardware is built around.
- `waitpad()` and `waitpadup` check for input in a loop that doesn't HALT at all, so the CPU will be maxed out until it returns. One alternative is to write a function with a loop that checks input with `joypad()` and then waits a frame using `wait_vbl_done()` (which idles the CPU while waiting) before checking input again.
- `joypad()`: When testing for multiple different buttons, it's best to read the joypad state *once* into a variable and then test using that variable (instead of making multiple calls).

5.3.5 Toolchain

- See SDCC optimizations: <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.8.1>
- Use profiling. Look at the ASM generated by the compiler, write several versions of a function, compare them and choose the faster one.
- Use the SDCC `--max-allocs-per-node` flag with large values, such as 50000. `--opt-code-speed` has a much smaller effect.
 - GBDK-2020 (after v4.0.1) compiles the library with `--max-allocs-per-node 50000`, but it must be turned on for your own code.
(example: `lcc ... -Wf--max-allocs-per-node50000` or `sdcc ... --max-allocs-per-node 50000`).

- The other code/speed flags are `--opt-code-speed` or `--opt-code-size`.
- Use current SDCC builds from <http://sdcc.sourceforge.net/snap.php>
The minimum required version of SDCC will depend on the GBDK-2020 release. See [GBDK Release Notes](#)
- Learn some ASM and inspect the compiler output to understand what the compiler is doing and how your code gets translated. This can help with writing better C code and with debugging.

5.3.6 Constants, Signed-ness and Overflows

There are a some scenarios where the compiler will warn about overflows with constants. They often have to do with mixed signedness between constants and variables. To avoid problems use care about whether or not constants are explicitly defined as unsigned and what type of variables they are used with.

WARNING: overflow in implicit constant conversion

- A constant can be used where the the value is too high (or low) for the storage medium causing an value overflow.
 - For example this constant value is too high since the max value for a signed 8 bit char is 127.
- This can also happen when constants are not explicitly declared as unsigned (and so may get treated by the compiler as signed) and then added such that the resulting value exceeds the signed maximum.
 - For example, this results in an warning even though the sum total is 254 which is less than the 255, the max value for a unsigned 8 bit char variable.

```
#define TOO_LARGE_CONST 255
int8_t signed_var = TOO_LARGE_CONST;
```

```
#define CONST_UNSIGNED 127u
#define CONST_SIGNED 127
uint8_t unsigned_var = (CONST_SIGNED + CONST_UNSIGNED);
```

- It can be avoided by always using the unsigned `u` when the constant is intended for unsigned operations.

```
#define CONST_UNSIGNED 127u
#define CONST_ALSO_UNSIGNED 127u // <-- Added "u", now no warning
uint8_t unsigned_var = (CONST_UNSIGNED + CONST_ALSO_UNSIGNED);
```

5.3.7 Chars and vararg functions

Parameters (chars, ints, etc) to `printf` / `sprintf` should always be explicitly cast to avoid type related parameter passing issues.

For example, below will result in the likely unintended output:

```
sprintf(str_temp, "%u, %d, %x\n", UINT16_MAX, INT16_MIN, UINT16_MAX);
printf("%s", str_temp);
// Will output: "65535, 0, 8000"
```

Instead this will give the intended output:

```
sprintf(str_temp, "%u, %d, %x\n", (uint16_t)UINT16_MAX, (int16_t)INT16_MIN, (uint16_t)UINT16_MAX);
printf("%s", str_temp);
// Will output: "65535, -32768, FFFF"
```

5.3.7.1 Chars In standard C when `chars` are passed to a function with variadic arguments (varargs, those declared with `...` as a parameter), such as `printf()`, those `chars` get automatically promoted to `ints`. For an 8 bit CPU such as the Game Boy's, this is not as efficient or desirable in most cases. So the default SDCC behavior, which GBDK-2020 expects, is that `chars` will remain `chars` and *not* get promoted to `ints` when **explicitly cast as chars while calling a varargs function**.

- They must be explicitly re-cast when passing them to a varargs function, even though they are already declared as `chars`.
- Discussion in SDCC manual:
 - <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.1.5>
 - <http://sdcc.sourceforge.net/doc/sdccman.pdf#subsection.3.5.10>
- If SDCC is invoked with `-std-cxx` (`-std-c89`, `-std-c99`, `-std-c11`, etc) then it will conform to standard C behavior and calling functions such as `printf()` with `chars` may not work as expected.

For example:

```
unsigned char i = 0x5A;
// NO:
// The char will get promoted to an int, producing incorrect printf output
// The output will be: 5A 00
printf("%hx %hx", i, i);
// YES:
// The char will remain a char and printf output will be as expected
// The output will be: 5A 5A
printf("%hx %hx", (unsigned char)i, (unsigned char)i);
```

Some functions that accept varargs:

- `EMU_printf`, `gprintf()`, `printf()`, `sprintf()`

Also See:

- Other cases of char to int promotion: <http://sdcc.sourceforge.net/doc/sdccman.pdf#chapter.6>

5.4 When C isn't fast enough

Todo Update and verify this section for the modernized SDCC and toolchain

For many applications C is fast enough but in intensive functions are sometimes better written in assembly. This section deals with interfacing your core C program with fast assembly sub routines.

5.4.1 Calling convention

SDCC in common with almost all C compilers prepends a `_` to any function names. For example the function `printf(...)` begins at the label `_printf:..` Note that all functions are declared global.

Functions can be marked with `OLDCALL` which will cause them to use the `__sdcccall(0)` calling convention (the format used prior to in SDCC 4.2 & GBDK-2020 4.1.0).

Starting with SDCC 4.2 and GBDK-2020 4.1.0 the new default calling convention is `__sdcccall(1)`.

For details about the calling conventions, see sections `SM83 calling conventions` and `Z80, Z180 and Z80N calling conventions` in the SDCC manual.

- <http://sdcc.sourceforge.net/doc/sdccman.pdf>

5.4.2 Variables and registers

Getting at C variables is slightly tricky due to how local variables are allocated on the stack. However you shouldn't be using the local variables of a calling function in any case. Global variables can be accessed by name by adding an underscore.

5.4.3 Segments / Areas

The use of segments/areas for code, data and variables is more noticeable in assembler. GBDK and SDCC define a number of default ones. The order they are linked is determined by `crt0.s` and is currently as follows for the Game Boy and related clones.

- ROM (in this order)
 - `_HEADER`: For the Game Boy header
 - `_CODE`: CODE is specified as after BASE, but is placed before it due to how the linker works.
 - `_HOME`
 - `_BASE`
 - `_CODE_0`
 - `_INITIALIZER`: Constant data used to init RAM data
 - `_LIT`
 - `_GSINIT`: Code used to init RAM data
 - `_GSFINAL`

- Banked ROM
 - `_CODE_x` Places code in ROM other than Bank 0, where x is the 16kB bank number.
- WRAM (in this order)
 - `_DATA`: Uninitialized RAM data
 - `_BSS`
 - `_INITIALIZED`: Initialized RAM data
 - `_HEAP`: placed after `_INITIALIZED` so that all spare memory is available for the malloc routines.
 - `STACK`: at the end of WRAM

6 ROM/RAM Banking and MBCs

6.1 ROM/RAM Banking and MBCs (Memory Bank Controllers)

The standard Game Boy cartridge with no MBC has a fixed 32K bytes of ROM. In order to make cartridges with larger ROM sizes (to store more code and graphics) MBCs can be used. They allow switching between multiple ROM banks that use the same memory region. Only one of the banks can be selected as active at a given time, while all the other banks are inactive (and so, inaccessible).

6.1.1 Non-banked cartridges

Cartridges with no MBC controller are non-banked, they have 32K bytes of fixed ROM space and no switchable banks. For these cartridges the ROM space between `0000h` and `7FFFh` can be treated as a single large bank of 32K bytes, or as two contiguous banks of 16K bytes in Bank 0 at `0000h` – `3FFFh` and Bank 1 at `4000h` to `7FFFh`.

6.1.2 MBC Banked cartridges (Memory Bank Controllers)

Cartridges with MBCs allow the the Game Boy to work with ROMs up to 8MB in size and with RAM up to 128kB. Each bank is 16K Bytes. The following are *usually* true, with some exceptions:

- Bank 0 of the ROM is located in the region at `0000h` – `3FFFh`. It is fixed (non-banked) and cannot be switched out for another bank.
- Banks 1 . . . N can be switched into the upper region at `4000h` – `7FFFh`. The upper limit for N is determined by the MBC used and available cartridge space.
- It is not necessary to manually assign Bank 0 for source files, that will happen by default if no bank is specified.

See the [Pandocs](#) for more details about the individual MBCs and their capabilities.

6.1.3 Recommended MBC type

For most projects we recommend **MBC5**.

- The `SWITCH_ROM()` / ref `SWITCH_RAM()` macros work with MBC5 (up to ROM bank 255, `SWITCH_ROM_MBC5_8M` may be used if a larger size is needed).
- **MBC1 is not recommended**. Some banks in it's range are unavailable. See pandocs for more details. <https://gbdev.io/pandocs/MBC1>

6.1.3.1 Bank 0 Size Limit and Overflows When Using MBCs When using MBCs and bank switching the space used in the lower fixed Bank 0 **must be \leq 16K bytes**. Otherwise it's data will overflow into Bank 1 and may be overwritten or overwrite other data, and can get switched out when banks are changed.

See the [FAQ entry about bank overflow errors](#).

6.1.3.2 Conserving Bank 0 for Important Functions and Data When using MBCs, Bank 0 is the only bank which is always active and its code can run regardless of what other banks are active. This means it is a limited resource and should be prioritized for data and functions which must be accessible regardless of which bank is currently active.

6.2 Working with Banks

To assign code and constant data (such as graphics) to a ROM bank and use it:

- Place the code for your ROM bank in one or several source files.
- Specify the ROM bank to use, either in the source file or at compile/link time.
- Specify the number of banks and MBC type during link time.
- When the program is running and wants to use data or call a function that is in a given bank, manually or automatically set the desired bank to active.

6.2.1 Setting the ROM bank for a Source file

The ROM and RAM bank for a source file can be set in a couple different ways. Multiple different banks cannot be assigned inside the same source file (unless the `__addressmod` method is used), but multiple source files can share the same bank.

If no ROM and RAM bank are specified for a file then the default `_CODE`, `_BSS` and `_DATA` segments are used.

Ways to set the ROM bank for a Source file:

- `#pragma bank <N>` at the start of a source file. Example (ROM bank 2): `#pragma bank 2`
- The lcc switch for ROM bank `-Wf-bo<N>`. Example (ROM bank 2): `-Wf-bo2`
- Using [rom_autobanking](#)

Note: You can use the `NONBANKED` keyword to define a function as non-banked if it resides in a source file which has been assigned a bank.

6.2.2 Setting the RAM bank for a Source file

- Using the lcc switch for Cartridge SRAM bank `-Wf-ba<N>`. Example (Cartridge SRAM bank 3): `-Wf-ba3`

6.2.3 Setting the MBC and number of ROM & RAM banks available

At the link stage this is done with `lcc` using pass-through switches for [makebin](#).

- `-Wm-yo<N>` where `<N>` is the number of ROM banks. 2, 4, 8, 16, 32, 64, 128, 256, 512
 - `-Wm-yoA` may be used for automatic bank size.
- `-Wm-ya<N>` where `<N>` is the number of RAM banks. 2, 4, 8, 16, 32
- `-Wm-yt<N>` where `<N>` is the type of MBC cartridge (see chart below).
 - Example: `Wm-yt0x1A`
- If passing the above arguments to [makebin](#) directly **without** using `lcc`, then the `-Wm` part should be omitted.
 - Note: Some makebin switches (such as `-yo A`) require a space when passed directly. See [makebin-settings](#) for details.

The MBC settings below are available when using the makebin `-Wl-yt<N>` switch.

Source: Pandocs. Additional details available at [Pandocs](#)

6.3 MBC Type Chart

```

0147: Cartridge type:
0x00: ROM ONLY
0x01: ROM+MBC1
0x02: ROM+MBC1+RAM
0x03: ROM+MBC1+RAM+BATT
0x05: ROM+MBC2
0x06: ROM+MBC2+BATTERY
0x08: ROM+RAM
0x09: ROM+RAM+BATTERY
0x0B: ROM+MMM01
0x0C: ROM+MMM01+SRAM
0x0D: ROM+MMM01+SRAM+BATT
0x0F: ROM+MBC3+TIMER+BATT
0x10: ROM+MBC3+TIMER+RAM+BATT
0x11: ROM+MBC3
0x12: ROM+MBC3+RAM
0x13: ROM+MBC3+RAM+BATT
0x19: ROM+MBC5
0x1A: ROM+MBC5+RAM
0x1B: ROM+MBC5+RAM+BATT
0x1C: ROM+MBC5+RUMBLE
0x1D: ROM+MBC5+RUMBLE+SRAM
0x1E: ROM+MBC5+RUMBLE+SRAM+BATT
0x1F: Pocket Camera
0xFD: Bandai TAMA5
0xFE: Hudson HuC-3
0xFF: Hudson HuC-1

```

Hex Code	MBC Type	SRAM	Battery	RTC	Rumble	Extra	Max ROM Size (1)
0x00	ROM ONLY						32 K
0x01	MBC-1 (2)						2 MB
0x02	MBC-1 (2)	SRAM					2 MB
0x03	MBC-1 (2)	SRAM	BATTERY				2 MB
0x05	MBC-2						256 K
0x06	MBC-2		BATTERY				256 K
0x08	ROM (3)	SRAM					32 K
0x09	ROM (3)	SRAM	BATTERY				32 K
0x0B	MMM01						8 MB / N
0x0C	MMM01	SRAM					8 MB / N
0x0D	MMM01	SRAM	BATTERY				8 MB / N
0x0F	MBC-3		BATTERY	RTC			2 MB
0x10	MBC-3 (4)	SRAM	BATTERY	RTC			2 MB
0x11	MBC-3						2 MB
0x12	MBC-3 (4)	SRAM					2 MB
0x13	MBC-3 (4)	SRAM	BATTERY				2 MB
0x19	MBC-5						8 MB
0x1A	MBC-5	SRAM					8 MB
0x1B	MBC-5	SRAM	BATTERY				8 MB
0x1C	MBC-5				RUMBLE		8 MB
0x1D	MBC-5	SRAM			RUMBLE		8 MB
0x1E	MBC-5	SRAM	BATTERY		RUMBLE		8 MB
0x20	MBC-6						~2MB
0x22	MBC-7	SRAM	BATTERY		RUMBLE	SENSOR	2MB
0xFC	POCKET CAMERA						To Do
0xFD	BANDAI TAMA5						To Do
0xFE	HuC3			RTC			To Do
0xFF	HuC1	SRAM	BATTERY			IR	To Do

1: Max possible size for MBC is shown. When used with generic [SWITCH_ROM\(\)](#) the max size may be smaller. For example:

- The max for MBC1 becomes **Bank 31** (512K)
- The max for MBC5 becomes **Bank 255** (4MB). To use the full 8MB size of MBC5 see [SWITCH_ROM_MBC5_8M\(\)](#).

2: For MBC1 some banks in it's range are unavailable. See pandocs for more details <https://gbdev.io/pandocs/MBC1>

3: No licensed cartridge makes use of this option. Exact behaviour is unknown.

4: MBC3 with RAM size 64 KByte refers to MBC30, used only in Pocket Monsters Crystal Version for Japan.

6.3.1 Getting Bank Numbers

The bank number for a banked function, variable or source file can be stored and retrieved using the following macros:

- [BANKREF\(\)](#): create a reference for retrieving the bank number of a variable or function
- [BANK\(\)](#): retrieve a bank number using a reference created with [BANKREF\(\)](#)
- [BANKREF_EXTERN\(\)](#): Make a [BANKREF\(\)](#) reference residing in another source file accessible in the current file for use with [BANK\(\)](#).

6.3.2 Banking and Functions

6.3.2.1 BANKED/NONBANKED Keywords for Functions

- **BANKED** (is a calling convention):
 - The function will use banked sdcc calls.
 - Placed in the bank selected by its source file (or compiler switches).
 - This keyword only specifies the **calling convention** for the function, it does not set a bank itself.
- **NONBANKED** (is a storage attribute):
 - Placed in the non-banked lower 16K region (bank 0), regardless of the bank selected by its source file.
 - Forces the `.area` to `_HOME`.
- `<not-specified>`:
 - The function does not use sdcc banked calls (`near` instead of `far`).
 - Placed in the bank selected by its source file (or compiler switches).

6.3.2.2 Banked Function Calls Banked functions can be called as follows:

- When defined with the **BANKED** keyword. Example: `void my_function() BANKED { do stuff }` in a source file which has had its bank set (see above).
- Using [far_pointers](#)
- When defined with an area set up using the `__addressmod` keyword (see the `banks_new` example project and the SDCC manual for details).
- Using [SWITCH_ROM\(\)](#) (and related functions for other MBCs) to manually switch in the required bank and then call the function.

Non-banked functions (either in fixed Bank 0, or in an non-banked ROM with no MBC):

- May call functions in any bank: **YES**
- May use data in any bank: **YES**

Banked functions (located in a switchable ROM bank)

- May call functions in fixed Bank 0: **YES**
- May call **BANKED** functions in any bank: **YES**
 - The compiler and library will manage the bank switching automatically using the bank switching trampoline.
- May use data in any bank: **NO**
 - May only use data from Bank 0 and the currently active bank.
 - A [NONBANKED wrapper function](#) may be used to access data in other banks.

Limitations:

- SDCC banked calls and `far_pointers` in GBDK only save one byte for the ROM bank. So, for example, they are limited to **bank 31** max for MBC1 and **bank 255** max for MBC5. This is due to the bank switching for those MBCs requiring a second, additional write to select the upper bits for more banks (banks 32+ in MBC1 and banks 256+ in MBC5).

6.3.3 Const Data (Variables in ROM)

Data declared as `const` (read only) will be stored in ROM in the bank associated with it's source file (if none is specified it defaults to Bank 0). If that bank is a switchable bank then the data is only accesible while the given bank is active.

6.3.4 Variables in RAM

Todo Variables in RAM

6.3.5 Far Pointers

Far pointers include a segment (bank) selector so they are able to point to addresses (functions or data) outside of the current bank (unlike normal pointers which are not bank-aware). A set of macros is provided by GBDK 2020 for working with far pointers.

Warning: Do not call the far pointer function macros from inside interrupt routines (ISRs). The far pointer function macros use a global variable that would not get restored properly if a function called that way was interrupted by another one called the same way. However, they may be called recursively.

See [FAR_CALL](#), [TO_FAR_PTR](#) and the `banks_farptr` example project.

6.3.6 Bank switching

You can manually switch banks using the [SWITCH_ROM\(\)](#), [SWITCH_RAM\(\)](#), and other related macros. See `banks.c` project for an example.

Note: You can only do a `switch_rom_bank` call from non-banked `__CODE` since otherwise you would switch out the code that was executing. Global routines that will be called without an expectation of bank switching should fit within the limited 16k of non-banked `__CODE`.

6.3.7 Wrapper Function for Accessing Banked Data

In order to load Data in one bank from code running in another bank a `NONBANKED` wrapper function can be used. It can save the current bank, switch to another bank, operate on some data, restore the original bank and then return.

An example function which can :

- Load background data from any bank
- And which can be called from code residing in any bank

```
// This function is NONBANKED so it resides in fixed Bank 0
void set_banked_bkg_data(uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data, uint8_t bank) NONBANKED
{
    uint8_t save = __current_bank;
    SWITCH_ROM(bank);
    set_bkg_data(first_tile, nb_tiles, data);
    SWITCH_ROM(save);
}
// And then it can be called from any bank:
set_banked_bkg_data(<first tile>, <num tiles>, tile_data, BANK(tile_data));
```

6.3.8 Currently active bank: `__current_bank`

The global variable `__current_bank` is updated automatically when calling [SWITCH_ROM\(\)](#), [SWITCH_ROM_MBC1\(\)](#) and [SWITCH_ROM_MBC5](#), or when a `BANKED` function is called.

Normaly banked calls are used and the active bank does not need to be directly managed, but in the case that it does the following shows how to save and restore it.

```
// The current bank can be saved
uint8_t _saved_bank = __current_bank;
// Call some function which changes the bank but does not restore it
// ...
// And then restored if needed
SWITCH_ROM(_saved_bank);
```

6.4 Auto-Banking

A ROM bank auto-assignment feature was added in GBDK 2020 4.0.2.

Instead of having to manually specify which bank a source file will reside in, the banks can be assigned automatically to make the best use of space. The bank assignment operates on object files, after compiling/assembling and before linking.

To turn on auto-banking, use the `-autobank` argument with `lcc`.

For a source example see the `banks_autobank` project.

In the source files you want auto-banked, do the following:

- Set the source file to be autobanked `#pragma bank 255` (this sets the temporary bank to 255, which `bankpack` then updates when repacking).
- Create a reference to store the bank number for that source file: `BANKREF (<some-bank-reference-name>)`.
 - More than one `BANKREF ()` may be created per file, but they should always have unique names.

In the other source files you want to access the banked data from, do the following:

- Create an extern so the bank reference in another file is accessible: `BANKREF_EXTERN (<some-bank-reference-name>)`.
- Obtain the bank number using `BANK (<some-bank-reference-name>)`.

Example: `level_1_map.c`

```
#pragma bank 255
BANKREF(level_1_map)
...
const uint8_t level_1_map[] = {... some map data here ...};
```

Accessing that data: `main.c`

```
BANKREF_EXTERN(level_1_map)
...
SWITCH_ROM( BANK(level_1_map) );
// Do something with level_1_map[]
```

Features and Notes:

- Fixed banked source files can be used in the same project as auto-banked source files. The `bankpack` tool will attempt to pack the auto-banked source files as efficiently as possible around the fixed-bank ones.

Making sure `bankpack` checks all files:

- In order to correctly calculate the bank for all files every time, it is best to use the `-ext=` flag and save the auto-banked output to a different extension (such as `.rel`) and then pass the modified files to the linker. That way all object files will be processed each time the program is compiled.

Recommended:

```
.c and .s -> (compiler) .o -> (bankpack) -> .rel -> (linker) ... -> .gb
```

- It is important because when `bankpack` assigns a bank for an autobanked (`bank=255`) object file (`.o`) it rewrites the bank and will then no longer see the file as one that needs to be auto-banked. That file will then remain in its previously assigned bank until a source change causes the compiler to rebuild it to an object file again which resets its bank to 255.
- For example consider a fixed-bank source file growing too large to share a bank with an auto-banked source file that was previously assigned to it. To avoid a bank overflow it would be important to have the auto-banked file check every time whether it can share that bank or not.
- See [bankpack](#) for more options and settings.

6.5 Errors related to banking (overflow, multiple writes to same location)

A *bank overflow* during compile/link time (in [makebin](#)) is when more code and data are allocated to a ROM bank than it has capacity for. The address for any overflowed data will be incorrect and the data is potentially unreachable since it now resides at the start of a different bank instead of the end of the expected bank.

See the [FAQ entry about bank overflow errors](#).

The current toolchain can only detect and warn (using [ihxcheck](#)) when one bank overflows into another bank that has data at its start. It cannot warn if a bank overflows into an empty one. For more complete detection, you can use the third-party [romusage](#) tool.

6.6 Bank space usage

In order to see how much space is used or remains available in a bank, you can use the third-party [romusage](#) tool.

6.6.1 Other important notes

- The [SWITCH_ROM_MBC5](#) macro is not interrupt-safe. If using less than 256 banks you may always use SWITCH_ROM - that is faster. Even if you use mbc5 hardware chip in the cart.

6.7 Banking example projects

There are several projects in the GBDK 2020 examples folder which demonstrate different ways to use banking.

- `Banks`: a basic banking example
- `Banks_new`: examples of using new bank assignment and calling conventions available in GBDK 2020 and its updated SDCC version.
- `Banks_farptr`: using far pointers which have the bank number built into the pointer.
- `Banks_autobank`: shows how to use the bank auto-assignment feature in GBDK 2020 4.0.2 or later, instead of having to manually specify which bank a source file will reside in.

7 GBDK Toolchain

7.1 Overview

GBDK 2020 uses the SDCC compiler along with some custom tools to build Game Boy ROMs.

- All tools are located under `bin/`
- The typical order of tools called is as follows (when using `lcc` these steps are usually performed automatically).
 1. Compile and assemble source files (`.c`, `.s`, `.asm`) with [sdcc](#) and [sdasgb](#)
 2. Optional: perform auto banking with [bankpack](#) on the object files
 3. Link the object files into `.ihx` file with [sdlldgb](#)
 4. Validate the `.ihx` file with [ihxcheck](#)
 5. Convert the `.ihx` file to a ROM file (`.gb`, `.gbc`) with [makebin](#)

To see individual arguments and options for a tool, run that tool from the command line with either no arguments or with `-h`.

7.2 Data Types

For data types and special C keywords, see [asm/sm83/types.h](#) and [asm/types.h](#).

Also see the SDCC manual (scroll down a little on the linked page): <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.1.1>

7.3 Changing Important Addresses

It is possible to change some of the important addresses used by the toolchain at link time using the `-Wl-g XXX=YYY` and `=Wl-b XXX=YYY` flags (where XXX is the name of the data, and YYY is the new address).

`lcc` will include the following linker defaults for [sdlldgb](#) if they are not defined by the user.

- `_shadow_OAM`
 - Location of sprite ram (requires 0xA0 bytes).
 - Default `-Wl-g _shadow_OAM=0xC000`
- `.STACK`

- Initial stack address
- Default `-Wl-g .STACK=0xE000`
- `.refresh_OAM`
 - Address to which the routine for refreshing OAM will be copied (must be in HIRAM). Default
 - Default `-Wl-g .refresh_OAM=0xFF80`
- `__DATA`
 - Start of RAM section (starts after Shadow OAM)
 - Default `-Wl-b __DATA=0xc0A0`
- `__CODE`
 - Start of ROM section
 - Default `-Wl-b __CODE=0x0200`

7.4 Compiling programs

The `lcc` program is the front end compiler driver for the actual compiler, assembler and linker. It works out what you want to do based on command line options and the extensions of the files you give it, computes the order in which the various programs must be called and then executes them in order. Some examples are:

- Compile the C source 'source.c', assemble and link it producing the Gameboy image 'image.gb'


```
lcc -o image.gb source.c
```
- Assemble the file 'source.s' and link it producing the Gameboy image 'image.gb'


```
lcc -o image.gb source.s
```
- Compile the C program 'source1.c' and assemble it producing the object file 'object1.o' for later linking.


```
lcc -c -o object1.o source1.c
```
- Assemble the file 'source2.s' producing the object file 'object2.o' for later linking


```
lcc -c -o object2.o source2.s
```
- Link the two object files 'object1.o' and 'object2.o' and produce the Gameboy image 'image.gb'


```
lcc -o image.gb object1.o object2.o
```
- Do all sorts of clever stuff by compiling then assembling source1.c, assembling source2.s and then linking them together to produce image.gb.


```
lcc -o image.gb source1.c source2.s
```

Arguments to the assembler, linker, etc can be passed via `lcc` using `-Wp...`, `-Wf...`, `-Wa...` and `-Wl...` to pass options to the pre-processor, compiler, assembler and linker respectively. Some common options are:

- To generate an assembler listing file.


```
-Wa-l
```
- To generate a linker map file.


```
-Wl-m
```
- To bind var to address 'addr' at link time.


```
-Wl-gvar=addr
```

For example, to compile the example in the memory section and to generate a listing and map file you would use the following. Note the leading underscore that C adds to symbol names.

```
lcc -Wa-l -Wl-m -Wl-g_snd_stat=0xff26 -o image.gb hardware.c
```

7.4.1 Makefiles

7.4.2 Using Makefiles

Please see the sample projects included with GBDK-2020 for a couple different examples of how to use Makefiles. You may also want to read a tutorial on Makefiles. For example:

<https://makefiletutorial.com/>
<https://www.tutorialspoint.com/makefile/index.htm>

7.4.3 Linker Files and ROM Auto Banking

When **bankpack** is called through **lcc** it will now always use linkerfile output (`-lkout=`) for passing files to the linker (all input object files and linkerfiles will get consolidated to a single linkerfile).

Bankpack:

- `lkin=<filename>` : Adds a input linkerfile (can specify multiple ones)
- `-lkout=<filename>` : Enables linkerfile output and sets name (only one can be specified). ALL loaded object files, both from the command line and any loaded from linkerfiles will have their names written to this single output.

LCC + Bankpack:

- `lcc` passes all input linkerfiles (from `-Wl-f<name>`) to **bankpack** (`-lkin=`)
- Linkerfile output is always used when `lcc` calls **bankpack** (`-lkout=`)
- A temporary file name is used for **bankpack** linkerfile output.
- `lcc` clears out the linker object file and linkerfile lists, then uses the single linkerfile generated by **bankpack**

Also see the `linkerfile` example project.

7.5 Build Tools

7.5.1 lcc

`lcc` is the compiler driver (front end) for the GBDK/sdcc toolchain.

For detailed settings see [lcc-settings](#)

It can be used to invoke all the tools needed for building a rom. If preferred, the individual tools can be called directly.

- the `-v` flag can be used to show the exact steps `lcc` executes for a build
- `lcc` can compile, link and generate a binary in a single pass: `lcc -o somerom.gb somesource.c`
- `lcc` now has a `-debug` flag that will turn on the following recommended flags for debugging
 - `--debug` for `sdcc` (`lcc` equiv: `-Wf-debug`)
 - `-y` enables `.cdb` output for `sdldgb` (`lcc` equiv: `-Wl-y`)
 - `-j` enables `.noi` output for `sdldgb` (`lcc` equiv: `-Wl-j`)

7.5.2 sdcc

SDCC C Source compiler.

For detailed settings see [sdcc-settings](#)

- Arguments can be passed to it through `lcc` using `-Wf-<argument>` and `-Wp-<argument>` (pre-processor)

7.5.3 sdasgb

SDCC Assembler for the Game Boy.

For detailed settings see [sdasgb-settings](#)

- Arguments can be passed to it through `lcc` using `-Wa-<argument>`

7.5.4 bankpack

Automatic Bank packer.

For detailed settings see [bankpack-settings](#)

When enabled, automatically assigns banks for object files where bank has been set to 255, see [rom_autobanking](#). Unless an alternative output is specified the given object files are updated with the new bank numbers.

- Can be enabled by using the `-autobank` argument with [lcc](#).
- Must be called after compiling/assembling and before linking.
- Arguments can be passed to it through [lcc](#) using `-Wb-<argument>`

7.5.5 sdldgb

The SDCC linker for the gameboy.

For detailed settings see [sdldgb-settings](#)

Links object files (.o) into a .ihx file which can be processed by [makebin](#)

- Arguments can be passed to it through [lcc](#) using `-Wl-<argument>`

7.5.6 ihxcheck

IHX file validator.

For detailed settings see [ihxcheck-settings](#)

Checks .ihx files produced by [sdldgb](#) for correctness.

- It will warn if there are multiple writes to the same ROM address. This may indicate mistakes in the code or ROM bank overflows
- Arguments can be passed to it through [lcc](#) using `-Wi-<argument>`

7.5.7 makebin

IHX to ROM converter.

- For detailed settings see [makebin-settings](#)
- For makebin `-yt` MBC values see [setting_mbc_and_rom_ram_banks](#)

Converts .ihx files produced by [sdldgb](#) into ROM files (.gb, .gbc). Also used for setting some ROM header data.

- Arguments can be passed to it through [lcc](#) using `-Wm-<argument>`

7.6 GBDK Utilities

7.6.1 GBCompress

Compression utility.

For detailed settings see [gbcompress-settings](#)

Compresses (and decompresses) binary file data with the gbcompress algorithm (also used in GBTD/GBMB). Decompression support is available in GBDK, see [gb_decompress\(\)](#).

Can also compress (and decompress) using block style RLE encoding with the `--alg=rle` flag. Decompression support is available in GBDK, see [rle_decompress\(\)](#).

7.6.2 png2asset

Tool for converting PNGs into GBDK format MetaSprites and Tile Maps.

- Convert single or multiple frames of graphics into metasprite structured data for use with the ...metasprite...() functions.
- When `-map` is used, converts images into Tile Maps and matching Tile Sets
- Supports Game Boy 2bpp, GBC 4bpp, SGB 4bpp, and SMS/GG 4bpp

For detailed settings see [png2asset-settings](#)

For working with sprite properties (including cgb palettes), see [metasprite_and_sprite_properties](#)

For API support see [move_metasprite\(\)](#) and related functions in [metasprites.h](#)

7.6.2.1 Working with png2asset

- The origin (pivot) for the metasprite is not required to be in the upper left-hand corner as with regular hardware sprites. See `-px` and `-py`.
- The conversion process supports using both `SPRITES_8x8` (`-spr8x8`) and `SPRITES_8x16` mode (`-spr8x16`). If 8x16 mode is used then the height of the metasprite must be a multiple of 16.

7.6.2.1.1 Terminology

The following abbreviations are used in this section:

- Original Game Boy and Game Boy Pocket style hardware: DMG
- Game Boy Color: CGB

7.6.2.1.2 Conversion Process

png2asset accepts any png as input, although that does not mean any image will be valid. The program will follow the next steps:

- The image will be subdivided into tiles of 8x8 or 8x16.
- For each tile a palette will be generated.
- If there are more than 4 colors in the palette it will throw an error.
- The palette will be sorted from darkest to lightest. If there is a transparent color that will be the first one (this will create a palette that will also work with DMG devices).
- If there are more than 8 palettes the program will throw an error.

With all this, the program will generate a new indexed image (with palette), where each 4 colors define a palette and all colors within a tile can only have colors from one of these palettes

It is also possible to pass a indexed 8-bit png with the palette properly sorted out, using `-keep_palette_order`

- Palettes will be extracted from the image palette in groups of 4 colors.
- Each tile can only have colors from one of these palettes per tile.
- The maximum number of colors is 32.

Using this image a tilesheet will be created

- Duplicated tiles will be removed.
- Tiles will be matched without mirror, using vertical mirror, horizontal mirror or both (use `-noflip` to turn off matching mirrored tiles).
- The palette won't be taken into account for matching, only the pixel color order, meaning there will be a match between tiles using different palettes but looking identical on grayscale.

7.6.2.1.3 Maps Passing `-map` the png can be converted to a map that can be used in both the background and the window. In this case, `png2asset` will generate:

- The palettes
- The tileset
- The map
- The color info
 - By default, an array of palette index for each tile. This is not the way the hardware works but it takes less space and will create maps compatibles with both DMG and CGB devices.
 - Passing `-use_map_attributes` will create an array of map attributes. It will also add mirroring info for each tile and because of that maps created with this won't be compatible with DMG.
 - * Use `-noflip` to make background maps which are compatible with DMG devices.

7.6.2.1.4 Meta sprites By default the png will be converted to metasprites. The image will be subdivided into meta sprites of `-sw x -sh`. In this case `png2asset` will generate:

- The metasprites, containing an array of:
 - tile index
 - y offset
 - x offset
 - flags, containing the mirror info, the palettes for both DMG and GBC and the sprite priority
- The metasprites array

7.6.2.1.5 Super Game Boy Borders (SGB) Screen border assets for the Super Game Boy can be generated using `png2asset`.

The following flags should be used to perform the conversion:

- `<input_border_file.png> -map -bpp 4 -max_palettes 4 -pack_mode sgb -use_map_attributes -c <output_border_data.c>`
- Where `<input_border_file.png>` is the image of the SGB border (256x224) and `<output_border_data.c>` is the name of the source file to write the assets out to.

See the `sgb_border` example project for more details.

7.6.3 makecom

Converts a binary `.rom` file to `.msxdos com` format, including splitting the banks up into separate files.

- For detailed settings see [makecom-settings](#)

8 Supported Consoles & Cross Compiling

8.1 Consoles Supported by GBDK

As of version 4.0.5 GBDK includes support for other consoles in addition to the Game Boy.

- Game Boy and related clones
 - Nintendo Game Boy / Game Boy Color (GB/GBC)
 - Analogue Pocket (AP)
 - Mega Duck / Cougar Boy (DUCK)
- Sega Consoles

- Sega Master System (SMS)
- Sega Game Gear (GG)
- MSX DOS (MSXDOS) (partial support)
- NES (NES) (partial support)

While the GBDK API has many convenience functions that work the same or similar across different consoles, it's important to keep their different capabilities in mind when writing code intended to run on more than one. Some (but not all) of the differences are screen sizes, color capabilities, memory layouts, processor type (z80 vs gbz80/sm83) and speed.

8.2 Cross Compiling for Different Consoles

8.2.1 lcc

When compiling and building through [lcc](#) use the `-m<port>:<plat>` flag to select the desired console via its port and platform combination.

8.2.2 sdcc

When building directly with the sdcc toolchain, the following must be specified manually (when using [lcc](#) it will populate these automatically based on `-m<port>:<plat>`).

When compiling with [sdcc](#):

- `-m<port>`, `-D__PORT_<port>` and `-D__TARGET_<plat>`

When assembling with [sdasgb](#) (for GB/AP) and [sdasz80](#) (for SMS/GG):

- Select the appropriate include path: `-I<gbdk-path>lib/<plat>`

When linking with [sldlgb](#) (for GB/AP) and [sldlz80](#) (for SMS/GG or MSXDOS):

- Select the appropriate include paths: `-k <gbdk-path>lib/<port>`, `-k <gbdk-path>lib/<plat>`
- Include the appropriate library files `-l <port>.lib`, `-l <plat>.lib`
- The crt will be under `<gbdk-path>lib/<plat>/crt0.o`

MSXDOS requires an additional build step with [makecom](#) after [makebin](#) to create the final binary:

- `makecom <image.bin> [<image.noi>] <output.com>`

The NES port has `--no-peep` specified (in [lcc](#)) due to a peephole related codegen bug in SDCC that has not yet been merged.

- If you wish to build without that flag then SDCC can be called directly instead of through [lcc](#).
- Alternately, custom peephole rules from a file can be passed in using `-Wf--peep-file (lcc)` or `--peep-file (sdcc)`.

8.2.3 Console Port and Platform Settings

Note: Starting with GBDK-2020 4.1.0 and SDCC 4.2, the Game Boy and related clones use `sm83` for the port instead of `gbz80`

- Nintendo Game Boy / Game Boy Color
 - [lcc](#): `-msm83:gb`
 - `port:sm83, plat:gb`
- Analogue Pocket

- `lcc` : `-msm83:ap`
 - `port:sm83, plat:ap`
- Mega Duck / Cougar Boy
 - `lcc` : `-msm83:duck`
 - `port:sm83, plat:duck`
- Sega Master System
 - `lcc` : `-mz80:sms`
 - `port:z80, plat:sms`
- Sega Game Gear
 - `lcc` : `-mz80:gg`
 - `port:z80, plat:gg`
- MSX DOS
 - `lcc` : `-mz80:msxdos`
 - `port:z80, plat:msxdos`
- NES
 - `lcc` : `-mos6502:nes`
 - `port:mos6502, plat:nes`

8.3 Cross-Platform Constants

There are several constant `#defines` that can be used to help select console specific code during compile time (with `#ifdef`, `#ifndef`).

8.3.1 Console Identifiers

- When `<gb/gb.h>` is included (either directly or through `<gbdk/platform.h>`)
 - When building for Game Boy:
 - * `NINTENDO` will be `#defined`
 - * `GAMEBOY` will be `#defined`
 - When building for Analogue Pocket
 - * `NINTENDO` will be `#defined`
 - * `ANALOGUEPOCKET` will be `#defined`
 - When building for Mega Duck / Cougar Boy
 - * `NINTENDO` will be `#defined`
 - * `MEGADUCK` will be `#defined`
- When `<sms/sms.h>` is included (either directly or through `<gbdk/platform.h>`)
 - When building for Master System
 - * `SEGA` will be `#defined`
 - * `MASTERSYSTEM` will be `#defined`
 - When building for Game Gear
 - * `SEGA` will be `#defined`
 - * `GAMEGEAR` will be `#defined`
- When `<msx/msx.h>` is included (either directly or through `<gbdk/platform.h>`)
 - `MSXDOS` will be `#defined`

8.3.2 Console Hardware Properties

Constants that describe properties of the console hardware are listed below. Their values will change to reflect the current console target that is being built.

- [DEVICE_SCREEN_X_OFFSET](#), [DEVICE_SCREEN_Y_OFFSET](#)
- [DEVICE_SCREEN_WIDTH](#), [DEVICE_SCREEN_HEIGHT](#)
- [DEVICE_SCREEN_BUFFER_WIDTH](#), [DEVICE_SCREEN_BUFFER_HEIGHT](#)
- [DEVICE_SCREEN_MAP_ENTRY_SIZE](#)
- [DEVICE_SPRITE_PX_OFFSET_X](#), [DEVICE_SPRITE_PX_OFFSET_Y](#)
- [DEVICE_SCREEN_PX_WIDTH](#), [DEVICE_SCREEN_PX_HEIGHT](#)

8.4 Using <gbdk/...> headers

Some include files under <gbdk/. . > are cross platform and others allow the build process to auto-select the correct include file for the current target port and platform (console). For example, the following can be used

```
#include <gbdk/platform.h>
#include <gbdk/metasprites.h>
```

Instead of

```
#include <gb/gb.h>
#include <gb/metasprites.h>
```

and

```
#include <sms/sms.h>
#include <sms/metasprites.h>
```

8.5 Cross Platform Example Projects

GBDK includes an number of cross platform example projects. These projects show how to write code that can be compiled and run on multiple different consoles (for example Game Boy and Game Gear) with, in some cases, minimal differences.

They also show how to build for multiple target consoles with a single build command and `Makefile`. The `Makefile.targets` allows selecting different `port` and `plat` settings when calling the build stages.

8.5.1 Cross Platform Asset Example

The cross-platform `Logo` example project shows how assets can be managed for multiple different console targets together.

In the example [utility_png2asset](#) is used to generate assets in the native format for each console at compile-time from separate source PNG images. The `Makefile` is set to use the source PNG folder which matches the current console being compiled, and the source code uses [set_native_tile_data\(\)](#) to load the assets tiles in native format.

8.6 Porting From Game Boy to Analogue Pocket

The Analogue Pocket operating in `.pocket` mode is (for practical purposes) functionally identical to the Game Boy / Color though it has a couple changes listed below. These are handled automatically in GBDK as long as the practices outlined below are followed.

Official differences:

- Altered register flag and address definitions
 - [STAT](#) & [LCDC](#): Order of register bits is reversed
 - * Example: [LCD on/off](#) is `LCDC.0` instead of `.7`
 - * Example: [LYC Interrupt enable](#) is `STAT.1` instead of `.6`

- [LCD](#) address is 0xFF4E instead of 0xFF40
- Different logo data in the header at address 0x0104:
 - 0x01, 0x10, 0xCE, 0xEF, 0x00, 0x00, 0x44, 0xAA, 0x00, 0x74, 0x00, 0x18, 0x11, 0x95, 0x00, 0x34, 0x00, 0x1A, 0x00, 0xD5, 0x00, 0x22, 0x00, 0x69, 0x6F, 0xF6, 0xF7, 0x73, 0x09, 0x90, 0xE1, 0x10, 0x44, 0x40, 0x9A, 0x90, 0xD5, 0xD0, 0x44, 0x30, 0xA9, 0x21, 0x5D, 0x48, 0x22, 0xE0, 0xF8, 0x60

Observed differences:

- MBC1 and MBC5 are supported, MBC3 won't save, the HuC3 isn't supported at all (via JoseJX)
- The Serial Link port does not work
- The IR port in CGB mode does not work as reliably as the Game Boy Color

In order for software to be easily ported to the Analogue Pocket, or to run on both, use the following practices.

8.6.1 Registers and Flags

Use API defined registers and register flags instead of hardwired ones.

- LCD register: [LCD_REG](#) or [rLCD](#)
- STAT register: [STAT_REG](#) or [rSTAT](#)
- LCD flags: -> [LCDCF_...](#) (example: [LCDCF_ON](#))
- STAT flags: -> [STATF_...](#) (example: [STATF_LYC](#))

8.6.2 Boot logo

As long as the target console is [set during build time](#) then the correct boot logo will be automatically selected.

8.7 Porting From Game Boy to Mega Duck / Cougar Boy

The Mega Duck is fairly similar to the classic Game Boy. It has a couple altered register flag and address definitions, no boot logo and a different startup/entry-point address. In order for software to be easily ported to the Mega Duck, or to run on both, use the following practices.

8.7.1 Registers and Flags

Use API defined registers and register flags instead of hardwired ones

- LCD register: [LCD_REG](#) or [rLCD](#)
- STAT register: [STAT_REG](#) or [rSTAT](#)
- LCD flags: -> [LCDCF_...](#) (example: [LCDCF_ON](#))
- STAT flags: -> [STATF_...](#) (example: [STATF_LYC](#))

8.8 Porting From Game Boy to SMS/GG

8.8.1 Tile Data and Tile Map loading

8.8.1.1 Tile and Map Data in 2bpp Game Boy Format

- [set_bkg_data\(\)](#) and [set_sprite_data\(\)](#) will load 2bpp tile data in "game boy" format on both GB and SMS/GG.
- On the SMS/GG [set_2bpp_palette\(\)](#) sets 4 colors that will be used when loading 2bpp assets with [set_bkg_data\(\)](#). This allows GB assets to be easily colorized without changing the asset format. There is some performance penalty for using the conversion.
- [set_bkg_tiles\(\)](#) loads 1-byte-per-tile tilemaps both for the GB and SMS/GG.

8.8.1.2 Tile and Map Data in Native Format Use the following api calls when assets are available in the native format for each platform.

[set_native_tile_data\(\)](#)

- GB/AP: loads 2bpp tiles data
- SMS/GG: loads 4bpp tile data

[set_tile_map\(\)](#)

- GB/AP: loads 1-byte-per-tile tilemaps
- SMS/GG: loads 2-byte-per-tile tilemaps

There are also bit-depth specific API calls:

- 1bpp: [set_1bpp_colors](#), [set_bkg_1bpp_data](#), [set_sprite_1bpp_data](#)
- 2bpp: [set_2bpp_palette](#), [set_bkg_2bpp_data](#), [set_sprite_2bpp_data](#), [set_tile_2bpp_data](#) (sms/gg only)
- 2bpp: [set_bkg_4bpp_data](#) (sms/gg only), [set_sprite_4bpp_data](#) (sms/gg only)

8.8.1.3 Emulated Game Boy Color map attributes on the SMS/Game Gear On the Game Boy Color, [VBK_REG](#) is used to select between the regular background tile map and the background attribute tile map (for setting tile color palette and other properties).

This behavior is emulated for the SMS/GG when using [set_bkg_tiles\(\)](#) and [VBK_REG](#). It allows writing a 1-byte tile map separately from a 1-byte attributes map.

Note

Tile map attributes on SMS/Game Gear use different control bits than the Game Boy Color, so a modified attribute map must be used.

8.9 Hardware Comparison

The specs below reflect the typical configuration of hardware when used with GBDK and is not meant as a complete list of their capabilities.

GB/AP

- Sprites:
 - 256 tiles (upper 128 are shared with background) (amount is doubled in CGB mode)
 - tile flipping/mirroring: yes
 - 40 total, max 10 per line
 - 2 x 4 color palette (color 0 transparent). 8 x 4 color palettes in CGB mode
- Background: 256 tiles (typical setup: upper 128 are shared with sprites) (amount is doubled in CGB mode)
 - tile flipping/mirroring: no (yes in CGB mode)
 - 1 x 4 color palette. 8 x 4 color palettes in CGB mode
- Window "layer": available
- Screen: 160 x 144
- Hardware Map: 256 x 256

SMS/GG

- Sprites:
 - 256 tiles (a bit less in the default setup)
 - tile flipping/mirroring: no
 - 64 total, max 8 per line

- 1 x 16 color palette (color 0 transparent)
- Background: 512 tiles (upper 256 are shared with sprites)
 - tile flipping/mirroring: yes
 - 2 x 16 color palettes
- Window "layer": not available
- SMS
 - Screen: 256 x 192
 - Hardware Map: 256 x 224
- GG
 - Screen: 160 x 144
 - Hardware Map: 256 x 224

8.9.1 Safe VRAM / Display Controller Access

GB/AP

- VRAM / Display Controller (PPU)
 - VRAM and some other display data / registers should only be written to when the [STATF_B_BUSY](#) bit of [STAT_REG](#) is off. Most GBDK API calls manage this automatically.

SMS/GG

- Display Controller (VDP)
 - Writing to the VDP should not be interrupted while an operation is already in progress (since that will interfere with the internal data pointer causing data to be written to the wrong location).
 - Recommended approach: Avoid writing to the VDP (tiles, map, scrolling, colors, etc) during an interrupt routine (ISR).
 - Alternative (requires careful implementation): Make sure writes to the VDP during an ISR are only performed when the [_shadow_OAM_OFF](#) flag indicates it is safe to do so.

9 Example Programs

GBDK includes several example programs both in C and in assembly. They are located in the examples directory, and in its subdirectories. They can be built by typing `make` in the corresponding directory.

9.1 banks (various projects)

There are several different projects showing how to use ROM banking with GBDK.

9.2 comm

Illustrates how to use communication routines.

9.3 crash

Demonstrates how to use the optional GBDK crash handler which dumps debug info to the Game Boy screen in the event of a program crash.

9.4 colorbar

The colorbar program, written by Mr. N.U. of TeamKNOx, illustrates the use of colors on a Color GameBoy.

9.5 dscan

Deep Scan is a game written by Mr. N.U. of TeamKNOx that supports the Color GameBoy. Your aim is to destroy the submarines from your boat, and to avoid the projectiles that they send to you. The game should be self-explanatory. The following keys are used:

```
RIGHT/LEFT : Move your boat
A/B         : Send a bomb from one side of your boat
START      : Start game or pause game
```

When game is paused:

```
SELECT      : Invert A and B buttons
RIGHT/LEFT  : Change speed
UP/DOWN     : Change level
```

9.6 filltest

Demonstrates various graphics routines.

9.7 fonts

Examples of how to work with the built in font and printing features.

9.8 galaxy

A C translation of the space.s assembly program.

9.9 gb-dtmf

The gb-dtmf, written by Osamu Ohashi, is a Dual Tone Multi-Frequency (DTMF) generator.

9.10 gbdecompress

Demonstrates using gbdecompress to load a compressed tile set into VRAM.

9.11 irq

Illustrates how to install interrupt handlers.

9.12 large map

Shows how to scroll with maps larger than 32 x 32 tiles using [set_bkg_submap\(\)](#). It fills rows and columns at the edges of the visible viewport (of the hardware Background Map) with the desired sub-region of the large map as it scrolls.

9.13 metasprites

Demonstrates using the metasprite features to move and animate a large sprite.

- Press A button to show / hide the metasprite
- Press B button to cycle through the metasprite animations
- Press SELECT button to cycle the metasprite through Normal / Flip-Y / Flip-XY / Flip-X
- Up / Down / Left / Right to move the metasprite

9.14 lcd isr wobble

An example of how to use the LCD ISR for visual special effects.

9.15 paint

The paint example is a painting program. It supports different painting tools, drawing modes, and colors. At the moment, it only paints individual pixels. This program illustrates the use of the full-screen drawing library. It also illustrates the use of generic structures and big sprites.

```
Arrow keys : Move the cursor
SELECT     : Display/hide the tools palette
A          : Select tool
```

9.16 rand

The rand program, written by Luc Van den Borre, illustrates the use of the GBDK random generator.

9.17 ram_fn

The ram_fn example illustrates how to copy functions to RAM or HIRAM, and how to call them from C.

9.18 rpnc

A basic RPN calculator. Try entering expressions like 12 134* and then 1789+.

9.19 samptest

Demonstration of playing a sound sample.

9.20 sgb (various)

A collection of examples showing how to use the Super Game Boy API features.

9.21 sound

The sound example is meant for experimenting with the sound generator of the GameBoy (to use on a real GameBoy). The four different sound modes of the GameBoy are available. It also demonstrates the use of bit fields in C (it's a quick hack, so don't expect too much from the code). The following keys are used:

```
UP/DOWN      : Move the cursor
RIGHT/LEFT   : Increment/decrement the value
RIGHT/LEFT+A : Increment/decrement the value by 10
RIGHT/LEFT+B : Set the value to maximum/minimum
START        : Play the current mode's sound (or all modes if in control screen)
START+A      : Play a little music with the current mode's sound
SELECT       : Change the sound mode (1, 2, 3, 4 and control)
SELECT+A     : Dump the sound registers to the screen
```

9.22 space

The space example is an assembly program that demonstrates the use of sprites, window, background, fixed-point values and more. The following keys are used:

```
Arrow keys    : Change the speed (and direction) of the sprite
Arrow keys + A : Change the speed (and direction) of the window
Arrow keys + B : Change the speed (and direction) of the background
START         : Open/close the door
SELECT        : Basic fading effect
```

9.23 templates

Two basic template examples are provided as a starting place for writing your GBDK programs.

10 Frequently Asked Questions (FAQ)

10.1 General

- How can sound effects be made?
 - The simplest way is to use the Game Boy sound hardware directly. See the [Sound Example](#) for a way to test out sounds on the hardware.
 - Further discussion on using the Sound Example rom can be found in the ZGB wiki. Note that some example code there is ZGB specific and not part of the base GBDK API: <https://github.com/Zal0/ZGB/wiki/Sounds>

10.2 Licensing

- What license information is required when distributing the compiled ROM (binary) of my game or program?
 - There is no requirement to include or credit any of the GBDK-2020 licenses or authors, although credit of GBDK-2020 is appreciated.
 - This is different and separate from redistributing the GBDK-2020 dev environment itself (or the GBDK-2020 sources) which does require the licenses.

10.3 Graphics and Resources

- How do I use a tile map when its tiles don't start at index zero?
 - The two main options are:
 - * Use [set_bkg_based_tiles\(\)](#), [set_bkg_based_submap\(\)](#), [set_win_based_tiles\(\)](#), [set_win_based_submap\(\)](#) and provide a tile origin offset.
 - * Use [utility_png2asset](#) with `-tile_origin` to create a map with the tile index offsets built in.

10.4 ROM Header Settings

- How do I set the ROM's title?
 - Use the [makebin](#) `-yn` flag. For example with [lcc](#) `-Wm-yn "MYTITLE"` or with [makebin](#) directly `-yn "MYTITLE"`. The maximum length is up to 15 characters, but may be shorter.
 - See "0134-0143 - Title" in [Pandocs](#) for more details.
- How do I set SGB, Color only and Color compatibility in the ROM header?
 - Use the following [makebin](#) flags. Prefix them with `-Wm` if using [lcc](#).
 - * `-yc` : GameBoy Color compatible
 - * `-yC` : GameBoy Color only
 - * `-ys` : Super GameBoy compatible
- How do I set the ROM [MBC](#) type, and what MBC values are available to use with the `-yt` [makebin](#) flag?
 - See [setting_mbc_and_rom_ram_banks](#)

10.5 Errors / Compiling / Toolchain

- What does the error `old "gbz80" SDCC PORT name specified (in "-mgbz80:gb"). Use "sm83" instead. You must update your build settings.` mean?
 - The PORT name for the Game Boy and related clones changed from `gbz80` to `sm83` in the SDCC version used in GBDK-2020 4.1.0 and later. You must change your Makefile, Build settings, etc to use the new name. Additional details in the [Console Port and Platform Settings](#) section.
- What does the warning `?ASlink-WARNING-Conflicting sdcc options: "-msm83" in module "_____" and "-mgbz80" in module "_____".` mean?
 - One object file was compiled with the PORT setting as `gbz80` (meaning a version of SDCC / GBDK-2020 **OLDER than GBDK-2020 4.1.0**).
 - The other had the PORT setting as `sm83` (meaning **GBDK-2020 4.1.0 or LATER**).
 - You must rebuild the object files using `sm83` with GBDK-2020 4.1.0 or later so that the linker is able to use them with the other object files. Additional details in the [Console Port and Platform Settings](#) section.
- What does `z80instructionSize() failed to parse line node, assuming 999 bytes` mean?
 - This is a known issue with SDCC Peephole Optimizer parsing and can be ignored. A bug report has been filed for it.
- What do these kinds of warnings / errors mean? `WARNING: possibly wrote twice at addr 4000 (93->3E) Warning: Write from one bank spans into the next. 7ff7 -> 8016 (bank 1 -> 2)`
 - You may have a overflow in one of your ROM banks. If there is more data allocated to a bank than it can hold it then will spill over into the next bank. The warnings are generated by [ihxcheck](#) during conversion of an `.ihx` file into a ROM file.
See the section [ROM/RAM Banking and MBCs](#) for more details about how banks work and what their size is. You may want to use a tool such as [romusage](#) to calculate the amount of free and used space.
- What does error: `size of the buffer is too small` mean?
 - Your program is using more banks than you have configured in the toolchain. Either the MBC type was not set, or the number of banks or MBC type should be changed to provide more banks.
See the section [setting_mbc_and_rom_ram_banks](#) for more details.
- What do the following kinds of warnings / errors mean? `info 218: z80instructionSize() failed to parse line node, assuming 999 bytes`
 - This is a known issue with SDCC, it should not cause actual problems and you can ignore the warning.
- Why is the compiler so slow, or why did it suddenly get much slower?
 - This may happen if you have large initialized arrays declared without the `const` keyword. It's important to use the `const` keyword for read-only data. See [const_gbtd_gbmb](#) and [const_array_data](#)
 - It can also happen if C source files are `#included` into other C source files, or if there is a very large source file.

- What flags should be enabled for debugging?
 - You can use the [lcc debug flag](#) `-debug` to turn on debug output. It covers most uses and removes the need to specify multiple flags such as `-Wa-l -Wl-m -Wl-j`.
- Is it possible to generate a debug symbol file (`.sym`) compatible with the [bgb](#) emulator?
 - Yes, turn on `.noi` output (LCC argument: `-Wl-j` or `-debug` and then use `-Wm-yS` with LCC (or `-yS` with `makebin` directly).
- How do I move the start of the `DATA` section and the `Shadow OAM` location?
 - The default locations are: `_shadow_OAM=0xC000` and 240 bytes after it `_DATA=0xC0A0`
 - So, for example, if you wanted to move them both to start 256(0x100) bytes later, use these command line arguments for LCC:
 - * To change the Shadow OAM address: `-Wl-g_shadow_OAM=0xC100`
 - * To change the DATA address (again, 240 bytes after the Shadow OAM): `-Wl-b_DATA=0xC1A0`
- What does this warning mean? `WARNING: overflow in implicit constant conversion`
 - See [Constants, Signed-ness and Overflows](#)

10.6 API / Utilities

- Is there a list of all functions in the API?
 - [Functions](#)
 - [Variables](#)
- Can I use the `float` type to do floating point math?
 - There is no support for 'float' in GBDK-2020.
 - Instead consider some form of `fixed` point math (including the [fixed](#) type included in GBDK).
- Why are 8 bit numbers not printing correctly with [printf\(\)](#)?
 - To correctly pass chars/uint8s for printing, they must be explicitly re-cast as such when calling the function. See [docs_chars_varargs](#) for more details.
- How can maps larger than 32x32 tiles be scrolled? & Why is the map wrapping around to the left side when setting a map wider than 32 tiles with [set_bkg_data\(\)](#)?
 - The hardware Background map is 32 x 32 tiles. The screen viewport that can be scrolled around that map is 20 x 18 tiles. In order to scroll around within a much larger map, new tiles must be loaded at the edges of the screen viewport in the direction that it is being scrolled. [set_bkg_submap](#) can be used to load those rows and columns of tiles from the desired sub-region of the large map.
 - See the "Large Map" example program and [set_bkg_submap\(\)](#).
 - Writes that exceed coordinate 31 of the Background tile map on the x or y axis will wrap around to the Left and Top edges.
- When using `gbt_player` with music in banks, how can the current bank be restored after calling `gbt_update()`? (since it changes the currently active bank without restoring it).

- See [restoring the current bank](#)
- How can CGB palettes and other sprite properties be used with metasprites?
 - See [Metasprites and sprite properties](#)
- Weird things are happening to my sprite colors when I use png2asset and metasprites. What's going on and how does it work?
 - See [utility_png2asset](#) for details of how the conversion process works.

11 Migrating to new GBDK Versions

This section contains information that may be useful to know or important when upgrading to a newer GBDK release.

11.1 GBDK-2020 versions

11.1.1 Porting to GBDK-2020 4.1.0

- GBDK now requires SDCC 4.2 or higher with GBDK-2020 patches for the the z80 linker
- The default calling convention changed in SDCC 4.2, see [Calling Conventions](#) for more details.
 - If you are linking to libraries compiled with an older version of SDCC / GBDK then you may have to recompile them.
 - If there are functions written in ASM which receive parameters, they should also be reviewed to make sure they work with the new `__sdcccall(1)` calling convention, or have their header declaration changed to use `OLDCALL`.
 - If you are using tools such as `rgb2sdas` (from hUGETracker/Driver) you may need to edit the resulting `.o` file and replace `-mgbz80` with `-msm83` in addition to using `OLDCALL`
- The SDCC `PORT` name for the Game Boy and related clones changed from `gbz80` to `sm83`.
 - Additional details in the [Console Port and Platform Settings](#) section and [FAQ entry](#). `lcc` will error out if the old `PORT` name is passed in.
- The library base path changed from `lib/small/asxxxx/` to `lib/`.
 - For example `lib/small/asxxxx/gb` becomes `lib/gb`
- Allocations for ISR chain lengths were fixed.
 - Now they are VBL: 4 user handlers, LCD: 3 user handlers, SIO/TIM/JOY: 4 user handlers

11.1.2 Porting to GBDK-2020 4.0.6

- Renamed `bgb_emu.h` to `emu_debug.h` and `BGB_*` functions to `EMU_*`
 - Aliases for the `BGB_*` ones and a `bgb_emu.h` shim are present for backward compatibility, but updating to the new naming is recommended

11.1.3 Porting to GBDK-2020 4.0.5

- GBDK now requires SDCC 12259 or higher with GBDK-2020 patches
- Variables in static storage are now initialized to zero per C standard (but remaining WRAM is not cleared)
- [png2asset](#) is the new name for the `png2mtspr` utility
- `lcc`: Changed default output format when not specified from `.ihx` to `.gb` (or other active rom extension)
- The `_BSS` area is deprecated (use `_DATA` instead)

- The `__BASE` area is renamed to `__HOME`
- Variables in static storage are now initialized to zero per C standard (but remaining WRAM is not cleared)
- `itoa()`, `uitoa()`, `ltoa()`, `ultoa()` all now require a radix value (base) argument to be passed. On the Game Boy and Analogue Pocket the parameter is required but not utilized.
- `set_bkg_1bit_data` has been renamed to `set_bkg_1bpp_data`
- The following header files which are now cross platform were moved from `gb/` to `gbdk/`: `bcd.h`, `console.h`, `far_ptr.h`, `font.h`, `gbdecompress.h`, `gbdk-lib.h`, `incbin.h`, `metasprites.h`, `platform.h`, `version.h`
 - When including them use `#include <gbdk/...>` instead of `#include <gb/>`

11.1.4 Porting to GBDK-2020 4.0.4

- GBDK now requires SDCC 12238 or higher
- Made `sample.h`, `cgb.h` and `sgb.h` independent from `gb.h`

11.1.5 Porting to GBDK-2020 4.0.3

- No significant changes required

11.1.6 Porting to GBDK-2020 4.0.2

- The default font has been reduced from 256 to 96 characters.
 - Code using special characters may need to be updated.
 - The off-by-1 character index offset was removed for fonts. Old fonts with the offset need to be re-adjusted.

11.1.7 Porting to GBDK-2020 4.0.1

- **Important!** : The WRAM memory region is no longer automatically initialized to zeros during startup.
 - Any variables which are declared without being initialized may have **indeterminate values instead of 0** on startup. This might reveal previously hidden bugs in your code.
 - Check your code for variables that are not initialized before use.
 - In BGB you can turn on triggering exceptions (options panel) reading from uninitialized RAM. This allows for some additional runtime detection of uninitialized vars.
- In `.ihx` files, multiple writes to the same ROM address are now warned about using `ihxcheck`.
- `set_*_tiles()` now wrap maps around horizontal and vertical boundaries correctly. Code relying on it not wrapping correctly may be affected.

11.1.8 Porting to GBDK-2020 4.0

- GBDK now requires SDCC 4.0.3 or higher
- The old linker `link-gbz80` has been REMOVED, the linker `sdlldgb` from SDCC is used.
 - Due to the linker change, there are no longer warnings about multiple writes to the same ROM address.
- GBDK now generates `.ihx` files, those are converted to a ROM using `makebin` (lcc can do this automatically in some use cases)
- Setting ROM bytes directly with `-Wl-yp0x<address>=0x<value>` is no longer supported. Instead use `makebin` flags. For example, use `-Wm-yC` instead of `-Wl-yp0x143=0xC0`. See [faq_gb_type_header_setting](#).
- OAM symbol has been renamed to `__shadow_OAM`, that allows accessing shadow OAM directly from C code

11.1.9 Porting to GBDK-2020 3.2

- No significant changes required

11.1.10 Porting to GBDK-2020 3.1.1

- No significant changes required

11.1.11 Porting to GBDK-2020 3.1

- Behavior formerly enabled by `USE_SFR_FOR_REG` is on by default now (no need to specify it, it isn't a tested `#ifdef` anymore). check here why: <https://gbdev.gg8.se/forums/viewtopic.php?id=697>

11.1.12 Porting to GBDK-2020 3.0.1

- LCC was upgraded to use SDCC v4.0. Makefile changes may be required
 - The symbol format changed. To get bgb compatible symbols turn on `.noi` output (LCC argument: `-Wl-j` or `-debug`) and use `-Wm-yS`
 - ?? Suggested: With LCC argument: `-Wa-l (sdasgb:-a All user symbols made global)`
 - In SDCC 3.6.0, the default for char changed from signed to unsigned.
 - * If you want the old behavior use `--fsigned-char`.
 - * lcc includes `--fsigned-char` by default
 - * Explicit declaration of unsigned vars is encouraged (for example, '15U' instead of '15')
 - `.init` address has been removed

11.2 Historical GBDK versions

11.2.1 GBDK 1.1 to GBDK 2.0

- Change your int variables to long if they have to be bigger than 255. If they should only contain values between 0 and 255, use an unsigned int.
- If your application uses the delay function, you'll have to adapt your delay values.
- Several functions have new names. In particular some of them have been changed to macros (e.g. `show_bkg()` is now `SHOW_BKG`).
- You will probably have to change the name of the header files that you include.

12 GBDK Release Notes

The GBDK-2020 releases can be found on Github: <https://github.com/gbdk-2020/gbdk-2020/releases>

12.1 GBDK-2020 Release Notes

12.1.1 GBDK-2020 4.1.0

2022/10

- Building GBDK
 - The linux port of SDCC is custom built on Ubuntu 16.04 due to reduced GLIBC compatibility issues in more recent SDCC project builds.
 - Added Windows 32-Bit build
- Platforms
 - SDCC has renamed the `gbz80` port to `sm83` see [faq_gbz80_sm83_old_port_name_error](#)

- Added experimental support for MSXDOS (`msxdos`) and NES (`nes`). These platforms are not fully functional at this time. See [Supported Consoles & Cross Compiling](#)
- Licensing
 - Clarified licensing status with consent from GBDK original authors, added licensing folder to distribution
- Library
 - SGB: Use longer wait between the SGB packet transfers
 - SMS/GG: less garbage on screen when clearing VRAM in the init code
 - SMS/GG: Added `cgb_compatibility()` to set default palette with the four shades of gray
 - Fixed: `get_sprite_data()`, `get_bkg_data()` , `get_win_data()` when `LDCDCF_BG8000` bit of `LDCDC_REG` is set
 - Fixed ISR chain lengths. VBL: 4 user handlers, LCD: 3 user handlers, SIO/TIM/JOY: 4 user handlers
 - Added new constants for the Game Boy Color (CGB):
 - * `VBK_BANK_0`, `VBK_BANK_1`
 - * `VBK_TILES`, `VBK_ATTRIBUTES`
 - * `BKGF_PRI`, `BKGF_YFLIP`, `BKGF_XFLIP`, `BKGF_BANK0`, `BKGF_BANK1`
 - * `BKGF_CGB_PAL0`, `BKGF_CGB_PAL1`, `BKGF_CGB_PAL2`, `BKGF_CGB_PAL3`, `BKGF_CGB_PAL4`, `BKGF_CGB_PAL5`, `BKGF_CGB_PAL6`, `BKGF_CGB_PAL7`
 - * `VBK_TILES`, `VBK_ATTRIBUTES`
- Toolchain / Utilities
 - `lcc`
 - * Changed to Error out and warn when `gbz80` port is used instead of `sm83`
 - `png2asset`
 - * Added `-tiles_only`: Export tile data only
 - * Added `-maps_only`: Export map tilemap only
 - * Added `-metasprites_only`: Export metasprite descriptors only
 - * Added `-source_tileset`: Use source tileset image with common tiles
 - * Added `-keep_duplicate_tiles`: Do not remove duplicate tiles
 - * Added `-bin`: Export to binary format (includes header files)
 - * Added `-transposed`: Export transposed (column-by-column instead of row-by-row)
 - * Added basic MSXDOS support
 - Added 1bpp packing mode (BPP1)
 - `-spr16x16msx`
 - * Added basic NES support
 - `-use_nes_attributes`
 - `-use_nes_colors`
 - * Changed to only export `_tile_pals[]` arrays when `-use-structs` is set (ZGB specific)
 - `gbcompress`
 - * Added `--bank=<num>` Add Bank Ref: 1 - 511 (default is none, with `--cout` only)
 - * Fixed failure to flush data at end of compression (uncommitted bytes)
 - * Fixed Warning: File read size didn't match expected
 - `lcc`
 - * When `-autobank` is specified `lcc` will automatically add `-yoA` for `makebin` if no `-yo*` entry is present
 - * Fixed broken `-E` Preprocess only flag
 - `makecom`
 - * Added `makecom` for post-processing `msxdos` binaries

- [makebin](#)
 - * Fixed (via `sdcc`) bug with `-yp` not always working correctly
 - <https://sourceforge.net/p/sdcc/code/12975/>
- [bankpack](#)
 - * Added support for the Game Boy Camera MBC
 - * Added `-reserve=<bank>:<size>` option to reserve space during autobank packing
 - Workaround for libraries that contain objects in banks (such as `gbt-player`)
- [ihxcheck](#)
 - * Check and warn for bank overflows under specific conditions
 - A multiple write to the same address must occur. The address where the overlap ends is used as BANK.
 - There must also be a write which spans multiple banks, the ending address of that must match BANK. The starting addresses is the OVERFLOW-FROM BANK.
- Examples
 - Changed Logo example to use new GBDK logo art from user "Digit"
 - Added example for APA image mode with more than 256 tiles
 - Added SGB Sound Effects example
 - Changed to new WAV sound example
- Docs
 - Added improved [MBC Type chart](#)
 - Include SDCC manual in pdf format
 - Various doc updates and improvements

12.1.2 GBDK-2020 4.0.6

2022/02

- Building GBDK
 - Changed to target older version of macOS (10.10) when building for better compatibility
- Platforms
 - Added support for Mega Duck / Cougar Boy (`duck`). See [Supported Consoles & Cross Compiling](#)
- Library
 - Added [memcmp\(\)](#)
 - Added [add_low_priority_TIM\(\)](#) function for timer interrupts which allow nesting for GB/CGB
 - Added [set_bkg_based_tiles\(\)](#), [set_bkg_based_submap\(\)](#), [set_win_based_tiles\(\)](#), [set_win_based_submap\(\)](#) for when a map's tiles don't start at VRAM index zero
 - Added [clock\(\)](#) for SMS/GG
 - Added macro definitions for SDCC features:
 - * `#define SFR __sfr`
 - * `#define AT(A) __at(A)`
 - Added check for OAM overflow to metasprite calls for GB/CGB
 - Added constant definitions [PSG_LATCH](#), [PSG_CH0](#), [PSG_CH1](#), [PSG_CH2](#), [PSG_CH3](#), [PSG_VOLUME](#) for SMS/GG
 - Renamed `bgb_emu.h` to `emu_debug.h` and `BGB_*` functions to `EMU_*`.

- * Aliases for the BGB_* ones and a bgb_emu.h shim are present for backward compatibility
 - Changed headers to wrap SDCC specific features (such as NONBANKED) with `#ifdef __SDCC`
 - Changed `rand()` and `arand()` to return `uint8_t` instead of `int8_t` (closer to the standard)
 - Fixed declaration for `PCM_SAMPLE` and definition for `AUD3WAVE`
 - Fixed definition of `size_t` to be `unsigned int` instead of `int`
 - Fixed `vmemcpy()` and `memmove()` for SMS/GG
 - Fixed random number generation for SMS/GG
 - Fixed letter U appearing as K for min font
 - Fixed define name in `crash_handler.h`
 - Exposed `__rand_seed`
- Toolchain / Utilities
 - `png2asset`
 - * Added SMS/GG graphics format support
 - * Added 4bpp and SGB borders
 - * Added warning when image size is not an even multiple of tile size
 - * Added `-tile_origin` offset option for when map tiles do not start at tile 0 in VRAM
 - * Added `*_TILE_COUNT` definition to output
 - * Fixed CGB `...s_map_attributes` type definition in output
 - * Fixed values for `num_palettes` in output
 - * Fixed incorrect `TILE_COUNT` value when not `-using_structs`
 - `lcc`
 - * Changed `makebin` flags to turn off Nintendo logo copy for GB/CGB (use version in crt instead)
 - * Fixed `lcc` handling of `makebin -x*` arguments
- Examples
 - Added logo example (cross-platform)
 - Added `ISR_VECTOR` example of a raw ISR vector with no dispatcher for GB/CGB
 - Changed `sgb_border` example to use `png2asset` for graphics
 - Changed use of `set_interrupts()` in examples so it's outside critical sections (since it disables/enables interrupts)
 - Changed cross-platform auto-banks example to use .h header files
 - Changed SGB border example to also work with SGB on PAL SNES
- Docs
 - Added new section: Migrating From Pre-GBDK-2020 Tutorials

12.1.3 GBDK-2020 4.0.5

2021/09

- Includes SDCC version 12539 with GBDK-2020 patches for Z80
- Known Issues
 - SDCC: `z80instructionSize()` failed to parse line node, assuming 999 bytes
 - * This is a known issue with the SDCC Peephole Optimizer parsing and can be ignored.
 - `-bo<n>` and `-ba<n>` are not supported by the Windows build of `sdcc`
 - On macOS the cross platform `banks` example has problems parsing the filename based ROM and RAM bank assignments into `-bo<n>` and `-ba<n>`

- Added support for new consoles. See [Supported Consoles & Cross Compiling](#)
 - Analogue Pocket (`ap`)
 - Sega Master System (`sms`) and Game Gear (`gg`)
- Library
 - Fixed error when calling `get_bkg_tile_xy`: '?ASlink-Warning-Undefined Global '.set_tile_xy' referenced by module '?ASlink-Warning-Byte PCR relocation error for symbol .set_tile_xy
 - Variables in static storage are now initialized to zero per C standard (but remaining WRAM is not cleared)
 - Added many new register flag constants and names. For example:
 - * `rLDC` is a new alias for `LDC_REG`
 - * `LDCF_WINON`, `LDCF_WINOFF`, `LDCF_B_WINON`
 - Added `BANK()`, `BANKREF()`, `BANKREF_EXTERN()`
 - Added `INCBIN()`, `BANK()`, `INCBIN_SIZE()`, `INCBIN_EXTERN()`
 - Added generic `SWITCH_ROM()` and `SWITCH_RAM()`
 - Added `BGB_printf()` and updated emulator debug output.
 - Added `set_native_tile_data()`, `set_tile_map()`, `set_1bpp_colors`, `set_bkg_1bpp_data`, `set_sprite_1bpp_data`, `set_2bpp_palette`, `set_bkg_2bpp_data`, `set_sprite_2bpp_data`, `set_tile_2bpp_data` (`sms/gg` only), `set_bkg_4bpp_data` (`sms/gg` only), `set_sprite_4bpp_data` (`sms/gg` only)
 - Added RLE decompression support: `rle_init()`, `rle_decompress()`,
 - Changed `itoa()`, `uitoa()`, `ltoa()`, `ultoa()` to now require a radix value (base) argument to be passed. On the Game Boy and Analogue Pocket the parameter is required but not utilized.
- Examples
 - Added cross-platform examples (build for multiple consoles: `gb`, `ap`, `sms`, `gg`)
 - Added `sms`, `gg`, `pocket(ap)` examples
 - Added `incbin` example
 - Added simple physics sub-pixel / fixed point math example
 - Added `rle` decompression example
 - Changed windows `make.bat` files to `compile.bat`
 - Bug fixes and updates for existing examples
- Toolchain / Utilities
 - `png2asset`
 - * `png2asset` is the new name for the `png2mtspr` utility
 - * Added collision rectangle width and height (`-pw`, `-ph`)
 - * Added option to use the palette from the source png (`-keep_palette_order`)
 - * Added option to disable tile flip (`-noflip`)
 - * Added export as map: `tileset + bg` (`-map`)
 - * Added option to use CGB BG Map attributes (`-use_map_attributes`)
 - * Added option to group the exported info into structs (`-use_structs`)
 - `lcc`
 - * Use `-m` to select target port and platform: `"-m[port]:[plat]"` ports: `gbz80`, `z80` plats: `ap`, `gb`, `sms`, `gg`
 - * Changed default output format when not specified from `.ihx` to `.gb` (or other active rom extension)
 - * Changed `lcc` to always use the linkerfile `-lkout=` option when calling `bankpack`
 - * Fixed name generation crash when outfile lacks extension
 - `bankpack`

- * Added linkerfile input and output: `-lkin=<file>`, `-lkout=<file>`
- * Added selector for platform specific behavior `plat=<plat>` (Default:gb, Available:gb, sms).
sms/gg targets prohibits packing `LIT_N` areas in the same banks as `CODE_N` areas
- * Added randomization for auto-banks (`-random`) for debugging and testing
- [utility_gbcompress](#)
 - * Added C source array format output (`-cout`) (optional variable name argument `-varname=`)
 - * Added C source array format input (`-cin`) (experimental)
 - * Added block style rle compression and decompression mode: `--alg=rle`
 - * Fixed compression errors when input size was larger than 64k
- Docs
 - Added [Supported Consoles & Cross Compiling](#) section
 - Various doc updates and improvements

12.1.4 GBDK-2020 4.0.4

2021/06

- Library
 - Support SDCC INITIALIZER area (SDCC ~12207+)
 - Added [get_vram_byte\(\)](#) / [get_win_tile_xy\(\)](#) / [get_bkg_tile_xy\(\)](#)
 - Added [set_tile_data\(\)](#)
 - Fixed SGB detection
 - Fixed broken [get_tiles\(\)](#) / [set_tiles\(\)](#)
 - Fixed broken token handling in [gb_decompress_sprite_data\(\)](#) / [gb_decompress_bkg_data\(\)](#) / [gb_decompress_win_data\(\)](#)
 - Changed all headers to use standard `stdint.h` types (ex: `uint8_t` instead of `UINT8/UBYTE`)
 - Made `sample.h`, `cgb.h` and `sgb.h` independent from `gb.h`
- Examples
 - Added project using a `.lk` linkerfile
 - Changed all examples to use standard `stdint.h` types
 - Moved `banks_farptr` and `banks_new` examples to "broken" due to SDCC changes
- Toolchain / Utilities
 - `png2mtspr`
 - * Added option to change default value for sprite property/attributes in (allows CGB palette, BG/WIN priority, etc).
 - * Improved: Turn off suppression of "blank" metasprite frames (composed of entirely transparent sprites)
 - * Fixed endless loop for png files taller than 255 pixels
 - `bankpack`
 - * Fixed `-yt mbc` specifier to also accept Decimal
 - * Improved: bank ID can be used in same file it is declared. Requires SDCC 12238+ with `-n` option to defer symbol resolution to link time.
 - `gbcompress`
 - * Added C source input (experimental) and output
 - * Added `size #defines`
 - `lcc`
 - * Added `-no-libs` and `-no-crt` options

- * Added support for .lk linker files (useful when number of files on lcc command line exceeds max size on windows)
 - * Added support for converting .ihx to .gb
 - * Added rewrite .o files -> .rel for linking when called with `-autobank` and `-Wb-ext=.rel`
 - * Workaround `makebin -Wl-yp` formatting segfault
- Docs
 - Improved `utility_png2mtspr` documentation
 - Various doc updates and improvements

12.1.5 GBDK-2020 4.0.3

2021/03

- Library
 - Added `set_vram_byte()`
 - Added `set_bkg_tile_xy()` / `set_win_tile_xy()`
 - Added `get_bkg_xy_addr()` / `get_win_xy_addr()`
 - Added `set_bkg_submap()` / `set_win_submap()`
 - Added metasprite api support
 - Added `gb_decompress` support
 - Added `calloc` / `malloc` / `realloc` / `free` and generic `memmove`
 - Improved `printf()`: ignore %0 padding and %1-9 width specifier instead of not printing, support upper case X
 - Fixed `line()`: handle drawing when x1 is less than x2
- Examples
 - Added `large_map`: showing how to use `set_bkg_submap()`
 - Added `scroller`: showing use of `get_bkg_xy_addr()`, `set_bkg_tile_xy()` and `set_vram_byte`
 - Added `gbdecompress`: de-compressing tile data into vram
 - Added `metasprites`: show creating a large sprite with the new metasprite api
 - Added `template` projects
 - Fixed build issue with `banks_autobank` example
 - Improved `sgb_border`
- Toolchain / Utilities
 - Added `utility_gbcompress` utility
 - Added `utility_png2mtspr` metasprite utility
- Docs
 - Added extensive documentation (some of which is imported and updated from the old gbdk docs)
 - Added PDF version of docs

12.1.6 GBDK-2020 4.0.2

2021/01/17

- Includes SDCC snapshot build version 12016 (has a fix for duplicate debug symbols generated from inlined header functions which GBDK 4.0+ uses)
- Updated documentation
- Library was improved
 - Linking with stdio.h does not require that much ROM now
 - Default font is changed to the smaller one (102 characters), that leaves space for user tiles
 - Fixed broken support for multiplying longs
 - memset/memcpy minor enhancements
 - safer copy-to-VRAM functions
 - loading of 1bit data fixed, also now it is possible to specify pixel color
 - Improved code generation for the GBDK Library with SDCC switch on by default: `--max-allocs-per-node 50000`
 - fixed wrong parameter offsets in `hiramcpy()` (broken `ram_function` example)
 - Multiple minor improvements
- New bankpack feature, allows automatic bank allocation for data and code, see `banks_autobank` example, feature is in beta state, use with care
- Lcc improvements
 - Fixed option to specify alternate base addresses for `shadow_OAM`, etc
- Examples: Added `bgb` debug example

12.1.7 GBDK-2020 4.0.1

2020/11/14

- Updated API documentation
- IHX is checked for correctness before the makebin stage. That allows to warn about overwriting the same ROM addresses (SDCC toolchain does not check this anymore).
- Library was improved
 - `set_*_tiles()` now wrap maps around horizontal and vertical boundaries correctly
 - new `fill_*_rect()` functions to clear rectangle areas
 - runtime initialization code now does not initialize whole WRAM with zeros anymore, that allows BGB to raise exceptions when code tries to read WRAM that was not written before.
 - enhanced SGB support
 - * `joypad_init()` / `joypad_ex()` support for multiple joypads
 - * SGB border example
 - `_current_bank` variable is updated when using bank switching macros
 - Reorganized examples: each example is in separate folder now, that simplifies understanding.
 - Lcc improvements
 - * Fix -S flag
 - * Fix default stack location from 0xDEFF to 0xE000 (end of WRAM1)
 - * Fix cleanup of .adb files with -Wf-debug flag
 - * Fix output not working if target is -o some_filename.ihx

12.1.8 GBDK-2020 4.0

2020/10/01

- GBDK now requires SDCC 4.0.3 or higher, that has fully working toolchain. Old link-gbz80 linker is not used anymore, sldlgb and makebin are used to link objects and produce binary roms; maccr tool is no longer needed either
 - SDCC 4.0.3 has much better code generator which produces smaller and faster code. Code is twice faster
 - SOURCE LEVEL DEBUGGING is possible now! Native toolchain produces *.CDB files that contain detailed debug info. Look for EMULICIOUS extension for vs.code. It supports breakpoints, watches, inspection of local variables, and more!
 - SDCC 4.0.4 has fixed RGBDS support; library is not updated to support that in full yet, but it is possible to assemble and link code emitted by SDCC with RGBDS
 - New banked trampolines are used, they are faster and smaller
 - New (old) initialization for non-constant arrays do NOT require 5 times larger rom space than initialized array itself, SDCC even tries to compress the data
- Library was improved
 - itoa/ltoa functions were rewritten, div/mod is not required now which is about 10 times faster
 - sprite functions are inline now, which is faster up to 12 times and produces the same or smaller code; .OAM symbol is renamed into _shadow_OAM that allows accessing shadow OAM directly from C code
 - interrupt handling was revised, it is now possible to make dedicated ISR's, that is important for time-sensitive handlers such as HBlank.
 - printf/sprintf were rewritten and splitted, print functions are twice faster now and also require less rom space if you use [sprintf\(\)](#) only, say, in bgb_emu.h
 - crash_handler.h - crash handler that allows to detect problems with ROMs after they are being released (adapted handler, originally written by ISSOtm)
 - improved and fixed string.h
 - many other improvements and fixes - thanks to all contributors!
- Revised examples
- Improved linux support
- Lcc has been updated
 - it works with the latest version of sdcc
 - quoted paths with spaces are working now

12.1.9 GBDK-2020 3.2

2020/06/05

- Fixed OAM initialization that was causing a bad access to VRAM
- Interrupt handlers now wait for lcd controller mode 0 or 1 by default to prevent access to inaccessible VRAM in several functions (like set_bkg_tiles)
- Several optimizations here and there

12.1.10 GBDK-2020 3.1.1

2020/05/17

- Fixed issues with libgcc_s_dw2-1.dll

12.1.11 GBDK-2020 3.1

2020/05/16

- Banked functions are working! The patcher is fully integrated in link-gbz80, no extra tools are needed. It is based on Toxa's work
 - Check this post for more info
 - Check the examples/gb/banked code for basic usage
- Behavior formerly enabled by USE_SFR_FOR_REG is on by default now (no need to specify it, it isn't a tested `#ifdef` anymore). check here why: <https://gbdev.gg8.se/forums/viewtopic.php?id=697>
- Fixed examples that were not compiling in the previous version and some improvements in a few of them. Removed all warnings caused by changing to the new SDCC
- Fixed bug in lcc that was causing some files in the temp folder not being deleted
- Removed as-gbz80 (the lib is now compiled with sdasgb thanks to this workaround) <https://github.com/gbdk-2020/gbdk-2020/commit/d2caafa4a66eb08998a14b258cb66af041a0e5c8>
- Profile support with bgb emulator
 - Basic support including `<gb/bgb_emu.h>` and using the macros `BGB_PROFILE_BEGIN` and `BGB_PROFILE_END`. More info in this post <https://gbdev.gg8.se/forums/viewtopic.php?id=703>
 - For full profiling check this repo and this post https://github.com/untoxa/bgb_profiling_toolkit/blob/master/readme.md <https://gbdev.gg8.se/forums/viewtopic.php?id=710>

12.1.12 GBDK-2020 3.0.1

2020/04/12

- Updated SDCC to v4.0
- Updated LCC to work with the new compiler

12.1.13 GBDK-2020 3.0

2020/04/12

- Initial GBDK-2020 release
Updated SDCC to v4.0 The new linker is not working so the old version is still there There is an issue with sdagb compiling drawing.s (the JP in line 32 after ".org .MODE_TABLE+4*.G_MODE" it's writing more than 4 bytes invading some addresses required by input.s:41) Because of this, all .s files in libc have been assembled with the old as-gbz80 and that's why it is still included

12.2 Historical GBDK Release Notes**12.2.1 GBDK 2.96**

17 April, 2000

Many changes.

- Code generated is now much more reliable and passes all of sdcc's regression suite.
- Added support for large sets of local variables (>127 bytes).
- Added full 32 bit long support.
- Still no floating pt support.

12.2.2 GBDK 2.95-3

19th August, 2000

- Stopped lcc with sdcc from leaking .cdb files all across /tmp.
- Optimised < and > for 16 bit variables.
- Added a new lexer to sdcc. Compiling files with large initialised arrays takes 31% of the time (well, at least samptest.c does :)

This is an experimental release for those who feel keen. The main change is a new lexer (the first part in the compilation process which recognises words and symbols like '!=' and 'char' and turns them into a token number) which speeds up compilation of large initialised arrays like tile data by a factor of three. Please report any bugs that show up - this is a big change.

I have also included a 'minimal' release for win32 users which omits the documentation, library sources, and examples. If this is useful I will keep doing it.

12.2.3 GBDK 2.95-2

5th August, 2000

Just a small update. From the README:

- Added model switching support —model-medium uses near (16 bit) pointers for data, and banked calls for anything not declared as 'nonbanked' —model-small uses near (16 bit) pointers for data and calls. Nothing uses banked calls. 'nonbanked' functions are still placed in HOME. Libraries are under lib/medium and lib/small.
- Added the gbdk version to 'sdcc -version'
- Changed the ways globals are exported, reducing the amount of extra junk linked in.
- Turned on the optimisations in flex. Large constant arrays like tile data should compile a bit faster.

12.2.4 GBDK 2.95

22nd July, 2000

- Fixed 'a << c' for c = [9..15]
- no\$gmb doesn't support labels of > 32 chars. The linker now trims all labels to 31 chars long.
- Fixed wait_vbl for the case where you miss a vbl
- Fixed + and - for any type where sizeof == 2 and one of the terms was on the stack. This includes pointers and ints. Fixes the text output bug in the examples. Should be faster now as well. Note that + and - for longs is still broken.
- Fixed the missing */ in gb.h
- Added basic far function support. Currently only works for isas and rgbasm. See examples/gb/far/*
- bc is now only pushed if the function uses it. i.e. something like: int silly(int i) { return i; } will not have the push bc; pop bc around it.
- Better rgbasm support. Basically:
 - o Use "sdcc -mgbz80 --asm=rgbds file.c" for each file.c
 - o Use "sdcc -mgbz80 --asm=rgbds crt0.o gbz80.lib gb.lib file1.o file2.o..."

to link everything together. The .lib files are generated using astorgb.pl and sdcc to turn the gbdk libraries into something rgbds compatible. The libraries are *not* fully tested. Trust nothing. But give it a go :)

- Ran a spell checker across the README and ChangeLog

This is a recommended upgrade. Some of the big features are:

Decent rgbds support. All the libraries and most of the examples can now compile with rgbds as the assembler. Banked function support. It is now easier to break the 32k barrier from within C. Functions can live in and be called transparently from any bank. Only works with rgbds Fixed some decent bugs with RSH, LSH, and a nasty bug with + and - for int's and pointers. Various optimisations in the code generator.

7th July, 2000

Information on float and long support. Someone asked about the state of float/long support recently. Heres my reply:

long support is partly there, as is float support. The compiler will correctly recognise the long and float keywords, and will generate the code for most basic ops (+, -, &, | etc) for longs correctly and will generate the function calls for floats and hard long operations (*, /, %) correctly. However it wont generate float constants in the correct format, nor will it 'return' a long or float - gbdk doesn't yet support returning types of 4 bytes. Unfortunately its not going to make it into 2.95 as there's too much else to do, but I should be able to complete long support for 2.96

12.2.5 GBDK 2.94

7th May, 2000

Many fixes - see the README for more.

7th May - Library documentation up. A good size part of the libraries that go with gbdk have been documented - follow the HTML link above to have a look. Thanks to quang for a good chunk of the gb.h documentation. Please report any errors :)

- Fixed #define BLAH 7 // Unterminated ' error in sdccpp
 - Fixed SCY_REG += 2, SCY_REG -= 5 (add and subtract in indirect space) as they were both quite broken.
 - externs and static's now work as expected.
 - You can now specify which bank code should be put into using a #pragma e.g: #pragma bank=HOME Under rgbds and asxxxx putting code in the HOME bank will force the code into bank 0 - useful for library functions. The most recent #pragma bank= will be the one used for the whole file.
 - Fixed an interesting bug in the caching of lit addresses
 - Added support for accessing high registers directly using the 'sfr' directive. See libc/gb/sfr.s and gb/hardware.h for an example. It should be possible with a bit of work to make high ram directly usable by the compiler; at the moment it is experimental. You can test sfr's by enabling USE_SFR_FOR_REGS=1
 - Added remove_VBL etc functions.
 - Documented the libs - see the gbdk-doc tarball distributed seperatly.
 - Two dimensional arrays seem to be broken.

12.2.6 GBDK 2.93

6th April, 2000

From the README

- Added multi-bank support into the compiler - The old -Wf-boxx and -Wf-baxx options now work
- Has preliminary support for generating rgbds and ISAS compatible assembler. Try -Wasm=rgbds or -Wasm=isas. The ISAS code is untested as I dont have access to the real assembler.
- RSH is fixed
- AND is fixed
- The missing parts of 2.1.0's libs are there. Note: They are untested.
- The dscan demo now fully works (with a hack :)
- There is a bug with cached computed values which are later used as pointers. When the value is first used as a BYTE arg, then later as a pointer the pointer fails as the high byte was never computed and is now missing. A temporary fix is to declare something appropriate as 'volatile' to stop the value being cached. See dscan.c/bombs() for an example.

12.2.7 GBDK 2.92-2 for win32

26th March, 2000

This is a maintenance release for win32 which fixes some of the niggly install problems, especially:

- win32 only. Takes care of some of the install bugs, including:
 - Now auto detects where it is installed. This can be overridden using set GBDKDIR=...
 - Problems with the installer (now uses WinZip)
 - Problems with the temp directory Now scans TMP, TEMP, TMPDIR and finally c: tmp
 - cygwin1.dll and 'make' are no longer required gbdk is now built using mingw32 which is win32 native make.bat is automatically generated from the Makefile
 - I've reverted to using WORD for signed 16 bit etc. GBDK_2_COMPAT is no longer required.

WORDS are now back to signed. GBDK_2_COMPAT is no longer needed. Temporary files are created in TMP, TEMP, or TMPDIR instead of c: tmp The installer is no more as it's not needed. There is a WinZip wrapped version for those with the extra bandwidth :). gbdk autodetects where it is installed - no more environment variables. cygwin1.dll and make are no longer required - gbdk is now compiled with mingw32.

See the ChangeLog section in the README for more information.

21st March, 2000

Problems with the installer. It seems that the demo of InstallVISE has an unreasonably short time limit. I had planed to use the demo until the license key came through, but there's no sign of the key yet and the 3 day evaluation is up. If anyone knows of a free Windows installer with the ability to modify environment variables, please contact me. I hear that temporarily setting you clock back to the 15th works...

18th March, 2000

libc5 version available / "Error creating temp file" Thanks to Rodrigo Couto there is now a Linux/libc5 version of gbdk3-2.92 available - follow the download link above. At least it will be there when the main sourceforge site comes back up... Also some people have reported a bug where the compiler reports '*** Error creating temp file'. Try typing "mkdir c: tmp" from a DOS prompt and see if that helps.

12.2.8 GBDK 2.92

8th March, 2000

Better than 2.91 :). Can now be installed anywhere. All the demos work. See the README for more.

- All the examples now work (with a little bit of patching :)
 - Fixed problem with registers being cached instead of being marked volatile.
 - More register packing - should be a bit faster.
 - You can now install somewhere except c: gbdk | /usr/lib/gbdk
 - Arrays initialised with constant addresses a'la galaxy.c now work.
 - Fixed minor bug with 104\$: labels in as.
 - Up to 167d/s...

12.2.9 GBDK 2.91

27th Feb, 2000

Better than 2.90 and includes Linux, win32 and a source tar ball. Some notes:

Read the README first Linux users need libgc-4 or above. Debian users try apt-get install libgc5. All the types have changed. Again, please read the README first. I prefer release early, release often. The idea is to get the bugs out there so that they can be squashed quickly. I've split up the libs so that they can be used on other platforms and so that the libs can be updated without updating the compiler. One side effect is that gb specific files have been shifted into their own directory i.e. gb.h is now gb/gb.h.

23rd Feb, 2000

First release of gbdk/sdcc. This is an early release - the only binary is for Linux and the source is only available through cvs. If your interested in the source, have a look at the cvs repository gbdk-support first, which will download all the rest of the code. Alternatively, look at gbdk-support and gbdk-lib at cvs.gbdk.sourceforge.net and sdcc at

cvs.sdcc.sourceforge.net. I will be working on binaries for Win32 and a source tar ball soon. Please report any bugs through the bugs link above.

31st Jan, 2000

Added Dermot's far pointer spec. It's mainly here for comment. If sdcc is ported to the Gameboy then I will be looking for some way to do far calls.

8th Jan, 2000

Moved over to sourceforge.net. Thanks must go to David Pfeffer for gbdk's previous resting place, www.gbdev.org. The transition is not complete, but cvs and web have been shifted. Note that the cvs download instructions are stale - you should now look to cvs.gbdk.sourceforge.net. I am currently working on porting sdcc over to the Z80. David Nathan is looking at porting it to the GB.

6th Jan, 2000

Icehawk wrote "I did write some rumble pack routines. Just make sure to remind people to add -Wl-yt0x1C or -Wl-yt0x1D or -Wl-yt0x1E depending on sram and battery usage. Find the routines on my site (as usual). =)"

18th Oct, 1999

Bug tracking / FAQ up. Try the link on the left to report any bugs with GBDK. It's also the first place to look if your having problems.

12.2.10 GBDK 2.1.5

17th Oct, 1999

The compiler is the same, but some of the libraries have been improved. [memset\(\)](#) and [memcpy\(\)](#) are much faster, [malloc\(\)](#) is fixed, and a high speed fixed block alternative [malloc\(\)](#) was added.

13 Toolchain settings

13.1 lcc settings

```
./lcc [ option | file ]...
    except for -l, options are processed left-to-right before files
    unrecognized options are taken to be linker options
-A warn about nonANSI usage; 2nd -A warns more
-b emit expression-level profiling code; see bprint(1)
-Bdir/ use the compiler named 'dir/rcc'
-c compile only
-dn set switch statement density to 'n'
-debug Turns on --debug for compiler, -y (.cdb) and -j (.noi) for linker
-Dname -Dname=def define the preprocessor symbol 'name'
-E only run preprocessor on named .c and .h files files -> stdout
--save-preproc Use with -E for output to *.i files instead of stdout
-g produce symbol table information for debuggers
-help or -? print this message
-Idir add 'dir' to the beginning of the list of #include directories
-K don't run ihxcheck test on linker ihx output
-lx search library 'x'
-m select port and platform: "-m[port]:[plat]" ports:sm83,z80,mos6502 plats:ap,duck,gb,sms,gg,nes
-N do not search the standard directories for #include files
-n emit code to check for dereferencing zero pointers
-no-crt do not auto-include the gbdk crt0.o runtime in linker list
-no-libs do not auto-include the gbdk libs in linker list
-O is ignored
-o file leave the output in 'file'
-P print ANSI-style declarations for globals
-p -pg emit profiling code; see prof(1) and gprof(1)
-S compile to assembly language
-autobank auto-assign banks set to 255 (bankpack)
-static specify static libraries (default is dynamic)
-t -tname emit function tracing calls to printf or to 'name'
-target name is ignored
-tempdir=dir place temporary files in 'dir/'; default=tmp
-Uname undefine the preprocessor symbol 'name'
-v show commands as they are executed; 2nd -v suppresses execution
-w suppress warnings
-Woarg specify system-specific 'arg'
-W[pfablmi]arg pass 'arg' to the preprocessor, compiler, assembler, bankpack, linker, ihxcheck, or makebin
```

13.2 sdcc settings

SDCC : z80/sm83/mos6502 4.2.2 #13350 (Linux)
published under GNU General Public License (GPL)
Usage : sdcc [options] filename
Options :-

```

General options:
--help                Display this help
-v                    Display sdcc's version
--verbose             Trace calls to the preprocessor, assembler, and linker
-V                    Execute verbosely. Show sub commands as they are run
-d                    Output list of macro definitions in effect. Use with -E
-D                    Define macro as in -Dmacro
-I                    Add to the include (*.h) path, as in -Ipath
-A
-U                    Undefine macro as in -Umacro
-M                    Preprocessor option
-W                    Pass through options to the pre-processor (p), assembler (a) or linker (l)
--include             Pre-include a file during pre-processing
-S                    Compile only; do not assemble or link
-c                    Compile and assemble, but do not link
-E                    Preprocess only, do not compile
--c1mode              Act in c1 mode. The standard input is preprocessed code, the output is assembly
                      code.
-o                    Place the output into the given path resp. file
-x                    Optional file type override (c, c-header or none), valid until the next -x

--print-search-dirs   display the directories in the compiler's search path
--vc                  messages are compatible with Microsoft visual studio
--use-stdout          send errors to stdout instead of stderr
--nostdlib            Do not include the standard library directory in the search path
--nostdinc            Do not include the standard include directory in the search path
--less-pedantic       Disable some of the more pedantic warnings
--disable-warning     <nnnn> Disable specific warning
--Werror              Treat the warnings as errors
--debug               Enable debugging symbol output
--cyclomatic          Display complexity of compiled functions
--std-c89              Use ISO C90 (aka ANSI C89) standard (slightly incomplete)
--std-sdcc89           Use ISO C90 (aka ANSI C89) standard with SDCC extensions
--std-c95              Use ISO C95 (aka ISO C94) standard (slightly incomplete)
--std-c99              Use ISO C99 standard (incomplete)
--std-sdcc99          Use ISO C99 standard with SDCC extensions
--std-c11              Use ISO C11 standard (incomplete)
--std-sdcc11           Use ISO C11 standard with SDCC extensions (default)
--std-c2x              Use ISO C2X standard (incomplete)
--std-sdcc2x           Use ISO C2X standard with SDCC extensions
--fdollars-in-identifiers Permit '$' as an identifier character
--fsigned-char         Make "char" signed by default
--use-non-free         Search / include non-free licensed libraries and header files

Code generation options:
-m                    Set the port to use e.g. -mz80.
-p                    Select port specific processor e.g. -mpic14 -p16f84

--stack-auto          Stack automatic variables
--xstack              Use external stack
--int-long-reent      Use reentrant calls on the int and long support functions
--float-reent         Use reentrant calls on the float support functions
--xram-movc           Use movc instead of movx to read xram (xdata)
--callee-saves        <func[,func,...]> Cause the called function to save registers instead of the
caller
--fomit-frame-pointer Leave out the frame pointer.
--all-callee-saves   callee will always save registers used
--stack-probe         insert call to function __stack_probe at each function prologue
--no-xinit-opt        don't memcpy initialized xram from code
--no-c-code-in-asm    don't include c-code as comments in the asm file
--no-peep-comments    don't include peephole optimizer comments
--codeseg             <name> use this name for the code segment
--constseg            <name> use this name for the const segment
--dataseg             <name> use this name for the data segment

Optimization options:
--nooverlay           Disable overlaying leaf function auto variables
--nogcse              Disable the GCSE optimisation
--nolabelopt          Disable label optimisation
--noinvariant         Disable optimisation of invariants
--noinduction         Disable loop variable induction
--noloopreverse       Disable the loop reverse optimisation
--no-peep             Disable the peephole assembly file optimisation
--no-reg-params       On some ports, disable passing some parameters in registers
--peep-asm            Enable peephole optimization on inline assembly
--peep-return         Enable peephole optimization for return instructions
--no-peep-return      Disable peephole optimization for return instructions
--peep-file           <file> use this extra peephole file
--opt-code-speed       Optimize for code speed rather than size
--opt-code-size        Optimize for code size rather than speed
--max-allocs-per-node Maximum number of register assignments considered at each node of the tree
decomposition
--nolospre            Disable lospre
--allow-unsafe-read    Allow optimizations to read any memory location anytime
--nostdlibcall        Disable optimization of calls to standard library

Internal debugging options:
--dump-ast            Dump front-end AST before generating i-code
--dump-i-code         Dump the i-code structure at all stages
--dump-graphs         Dump graphs (control-flow, conflict, etc)
--i-code-in-asm       Include i-code as comments in the asm file
--fverbose-asm        Include code generator comments in the asm output

```

```

Linker options:
-l          Include the given library in the link
-L          Add the next field to the library search path
--lib-path  <path> use this path to search for libraries
--out-fmt-ihx      Output in Intel hex format
--out-fmt-s19      Output in S19 hex format
--xram-loc         <nnnn> External Ram start location
--xram-size        <nnnn> External Ram size
--iram-size        <nnnn> Internal Ram size
--xstack-loc       <nnnn> External Stack start location
--code-loc         <nnnn> Code Segment Location
--code-size        <nnnn> Code Segment size
--stack-loc        <nnnn> Stack pointer initial value
--data-loc         <nnnn> Direct data start location
--idata-loc
--no-optsdcc-in-asm Do not emit .optsdcc in asm
Special options for the z80 port:
--callee-saves-bc   Force a called function to always save BC
--portmode=         Determine PORT I/O mode (z80/z180)
--bo                <num> use code bank <num>
--ba                <num> use data bank <num>
--asm=              Define assembler name (rgbds/asxxxx/isas/z80asm/gas)
--codeseg           <name> use this name for the code segment
--constseg          <name> use this name for the const segment
--dataseg           <name> use this name for the data segment
--no-std-crt0        Do not link default crt0.rel
--reserve-regs-iy    Do not use IY (incompatible with --fomit-frame-pointer)
--fno-omit-frame-pointer Do not omit frame pointer
--emit-externs       Emit externs list in generated asm
--legacy-banking     Use legacy method to call banked functions
--nmios-z80          Generate workaround for NMOS Z80 when saving IFF2
--sdcccall           Set ABI version for default calling convention
--allow-undocumented-instructions Allow use of undocumented instructions
Special options for the sm83 port:
--bo                <num> use code bank <num>
--ba                <num> use data bank <num>
--asm=              Define assembler name (rgbds/asxxxx/isas/z80asm/gas)
--callee-saves-bc   Force a called function to always save BC
--codeseg           <name> use this name for the code segment
--constseg          <name> use this name for the const segment
--dataseg           <name> use this name for the data segment
--no-std-crt0        Do not link default crt0.rel
--legacy-banking     Use legacy method to call banked functions
--sdcccall           Set ABI version for default calling convention
Special options for the mos6502 port:
--model-small       8-bit address space for data
--model-large        16-bit address space for data (default)
--no-std-crt0        Do not link default crt0.rel

```

13.3 sdasgb settings

```

sdas Assembler V02.00 + NoICE + SDCC mods (GameBoy)
Copyright (C) 2012 Alan R. Baldwin
This program comes with ABSOLUTELY NO WARRANTY.
Usage: [-Options] [-Option with arg] file
Usage: [-Options] [-Option with arg] outfile file1 [file2 ...]
-h or NO ARGUMENTS Show this help list
Input:
-I Add the named directory to the include file
  search path. This option may be used more than once.
  Directories are searched in the order given.
Output:
-l Create list file/outfile[.lst]
-o Create object file/outfile[.rel]
-s Create symbol file/outfile[.sym]
Listing:
-d Decimal listing
-q Octal listing
-x Hex listing (default)
-b Display .define substitutions in listing
-bb and display without .define substitutions
-c Disable instruction cycle count in listing
-f Flag relocatable references by ' in listing file
-ff Flag relocatable references by mode in listing file
-p Disable automatic listing pagination
-u Disable .list/.nlist processing
-w Wide listing format for symbol table
Assembly:
-v Enable out of range signed / unsigned errors
Symbols:
-a All user symbols made global
-g Undefined symbols made global
-n Don't resolve global assigned value symbols
-z Disable case sensitivity for symbols
Debugging:
-j Enable NoICE Debug Symbols

```

-y Enable SDCC Debug Symbols

13.4 sdasz80 settings

sdas Assembler V02.00 + NoICE + SDCC mods (GameBoy)
 Copyright (C) 2012 Alan R. Baldwin
 This program comes with ABSOLUTELY NO WARRANTY.
 Usage: [-Options] [-Option with arg] file
 Usage: [-Options] [-Option with arg] outfile file1 [file2 ...]
 -h or NO ARGUMENTS Show this help list
 Input:
 -I Add the named directory to the include file
 search path. This option may be used more than once.
 Directories are searched in the order given.
 Output:
 -l Create list file/outfile[.lst]
 -o Create object file/outfile[.rel]
 -s Create symbol file/outfile[.sym]
 Listing:
 -d Decimal listing
 -q Octal listing
 -x Hex listing (default)
 -b Display .define substitutions in listing
 -bb and display without .define substitutions
 -c Disable instruction cycle count in listing
 -f Flag relocatable references by ' in listing file
 -ff Flag relocatable references by mode in listing file
 -p Disable automatic listing pagination
 -u Disable .list/.nlist processing
 -w Wide listing format for symbol table
 Assembly:
 -v Enable out of range signed / unsigned errors
 Symbols:
 -a All user symbols made global
 -g Undefined symbols made global
 -n Don't resolve global assigned value symbols
 -z Disable case sensitivity for symbols
 Debugging:
 -j Enable NoICE Debug Symbols
 -y Enable SDCC Debug Symbols

13.5 bankpack settings

bankalloc [options] objfile1 objfile2 etc
 Use: Read .o files and auto-assign areas with bank=255.
 Typically called by Lcc compiler driver before linker.
 Options
 -h : Show this help
 -lkin=<file> : Load object files specified in linker file <file>
 -lkout=<file> : Write list of object files out to linker file <file>
 -yt<mbctype> : Set MBC type per ROM byte 149 in Decimal or Hex (0xNN)
 ([see pandocs] (https://gbdev.io/pandocs/The_Cartridge_Header.html#0147---cartridge-type))
 -mbc=N : Similar to -yt, but sets MBC type directly to N instead
 of by interpreting ROM byte 149
 mbc1 will exclude banks {0x20,0x40,0x60} max=127,
 mbc2 max=15, mbc3 max=127, mbc5 max=255 (not 511!)
 -min=N : Min assigned ROM bank is N (default 1)
 -max=N : Max assigned ROM bank is N, error if exceeded
 -ext=<.ext> : Write files out with <.ext> instead of source extension
 -path=<path> : Write files out to <path> (<path> *MUST* already exist)
 -sym=<prefix> : Add symbols starting with <prefix> to match + update list.
 Default entry is "__bank_" (see below)
 -cartsize : Print min required cart size as "autocartsize:<NNN>"
 -plat=<plat> : Select platform specific behavior (default:gb) (gb,sms)
 -random : Distribute banks randomly for testing (honors -min/-max)
 -reserve=<b:n>: Reserve N bytes (hex) in bank B (decimal)
 Ex: -reserve=105:30F reserves 0x30F bytes in bank 105
 -v : Verbose output, show assignments
 Example: "bankpack -ext=.rel -path=some/newpath/ file1.o file2.o"
 Unless -ext or -path specify otherwise, input files are overwritten.
 Default MBC type is not set. It *must* be specified by -mbc= or -yt!
 The following will have FF and 255 replaced with the assigned bank:
 A _CODE_255 size <size> flags <flags> addr <address>
 S b_<function name> Def0000FF
 S __bank_<const name> Def0000FF
 (Above can be made by: const void __at(255) __bank_<const name>;

13.6 sldlgb settings

sldl Linker V03.00 + NoICE + sldl
 Usage: [-Options] [-Option with arg] file
 Usage: [-Options] [-Option with arg] outfile file1 [file2 ...]
 Startup:

```

-p    Echo commands to stdout (default)
-n    No echo of commands to stdout
Alternates to Command Line Input:
-c          ASlink » prompt input
-f file[.lk] Command File input
Libraries:
-k    Library path specification, one per -k
-l    Library file specification, one per -l
Relocation:
-b    area base address = expression
-g    global symbol = expression
Map format:
-m    Map output generated as (out)file[.map]
-w    Wide listing format for map file
-x    Hexadecimal (default)
-d    Decimal
-q    Octal
Output:
-i    Intel Hex as (out)file[.ihx]
-s    Motorola S Record as (out)file[.s19]
-j    NoICE Debug output as (out)file[.noi]
-y    SDCDB Debug output as (out)file[.cdb]
List:
-u    Update listing file(s) with link data as file(s)[.rst]
Case Sensitivity:
-z    Disable Case Sensitivity for Symbols
End:
-e    or null line terminates input

```

13.7 sldz80 settings

```

sld Linker V03.00 + NoICE + sld
Usage: [-Options] [-Option with arg] file
Usage: [-Options] [-Option with arg] outfile file1 [file2 ...]
Startup:
-p    Echo commands to stdout (default)
-n    No echo of commands to stdout
Alternates to Command Line Input:
-c          ASlink » prompt input
-f file[.lk] Command File input
Libraries:
-k    Library path specification, one per -k
-l    Library file specification, one per -l
Relocation:
-b    area base address = expression
-g    global symbol = expression
Map format:
-m    Map output generated as (out)file[.map]
-w    Wide listing format for map file
-x    Hexadecimal (default)
-d    Decimal
-q    Octal
Output:
-i    Intel Hex as (out)file[.ihx]
-s    Motorola S Record as (out)file[.s19]
-j    NoICE Debug output as (out)file[.noi]
-y    SDCDB Debug output as (out)file[.cdb]
List:
-u    Update listing file(s) with link data as file(s)[.rst]
Case Sensitivity:
-z    Disable Case Sensitivity for Symbols
End:
-e    or null line terminates input

```

13.8 ihxcheck settings

```

ihx_check input_file.ihx [options]
Options
-h : Show this help
-e : Treat warnings as errors
Use: Read a .ihx and warn about overlapped areas.
Example: "ihx_check build/MyProject.ihx"

```

13.9 makebin settings

Also see [setting_mbc_and_rom_ram_banks](#)

makebin: convert a Intel IHX file to binary or GameBoy format binary.

Usage: makebin [options] [<in_file> [<out_file>]]

```

Options:
-p          pack mode: the binary file size will be truncated to the last occupied byte
-s romsize  size of the binary file (default: rom banks * 16384)
-Z          generate GameBoy format binary file

```



```

-S            generate Sega Master System format binary file
-N            generate Famicom/NES format binary file
-o bytes      skip amount of bytes in binary file
SMS format options (applicable only with -S option):
-xo n         rom size (0xa-0x2) (default: 0xc)
-xj n         set region code (3-7) (default: 4)
-xv n         version number (0-15) (default: 0)
GameBoy format options (applicable only with -Z option):
-yo n         number of rom banks (default: 2) (autosize: A)
-ya n         number of ram banks (default: 0)
-yt n         MBC type (default: no MBC)
-yl n         old licensee code (default: 0x33)
-yk cc        new licensee string (default: 00)
-yn name      cartridge name (default: none)
-yc           GameBoy Color compatible
-yC           GameBoy Color only
-ys           Super GameBoy
-yS           Convert .noi file named like input file to .sym
-yj           set non-Japanese region flag
-yN           do not copy big N validation logo into ROM header
-yp addr=value Set address in ROM to given value (address 0x100-0x1FE)
Arguments:
<in_file>     optional IHX input file, '-' means stdin. (default: stdin)
<out_file>     optional output file, '-' means stdout. (default: stdout)

```

13.10 makecom settings

makecom image.rom image.noi output.com
 Use: convert a binary .rom file to .msxdos com format.

13.11 gbcompress settings

```

gbcompress [options] infile outfile
Use: compress a binary file and write it out.
Options
-h          : Show this help screen
-d          : Decompress (default is compress)
-v          : Verbose output
--cin       : Read input as .c source format (8 bit char ONLY, uses first array found)
--cout      : Write output in .c / .h source format (8 bit char ONLY)
--varname=<NAME> : specify variable name for c source output
--alg=<type>    : specify compression type: 'rle', 'gb' (default)
--bank=<num>    : Add Bank Ref: 1 - 511 (default is none, with --cout only)
Example: "gbcompress binaryfile.bin compressed.bin"
Example: "gbcompress -d compressedfile.bin decompressed.bin"
Example: "gbcompress --alg=rle binaryfile.bin compressed.bin"
The default compression (gb) is the type used by gbtd/gbmb
The rle compression is Amiga IFF style

```

13.12 png2asset settings

```

usage: png2asset <file>.png [options]
-c          output file (default: <png file>.c)
-sw <width>  metasprites width size (default: png width)
-sh <height> metasprites height size (default: png height)
-sp <props>  change default for sprite OAM property bytes (in hex) (default: 0x00)
-px <x coord> metasprites pivot x coordinate (default: metasprites width / 2)
-py <y coord> metasprites pivot y coordinate (default: metasprites height / 2)
-pw <width>  metasprites collision rect width (default: metasprites width)
-ph <height> metasprites collision rect height (default: metasprites height)
-spr8x8     use SPRITES_8x8
-spr8x16    use SPRITES_8x16 (this is the default)
-spr16x16msx use SPRITES_16x16
-b <bank>    bank (default 0)
-keep_palette_order use png palette
-noflip      disable tile flip
-map         Export as map (tileset + bg)
-use_map_attributes Use CGB BG Map attributes
-use_nes_attributes Use NES BG Map attributes
-use_nes_colors   Convert RGB color values to NES PPU colors
-use_structs      Group the exported info into structs (default: false) (used by ZGB Game Engine)
-bpp            bits per pixel: 1, 2, 4 (default: 2)
-max_palettes    max number of palettes allowed (default: 8)
                (note: max colors = max_palettes x num colors per palette)
-pack_mode       gb, sgb, sms, lbpp (default: gb)
-tile_origin     tile index offset for maps (default: 0)
-tiles_only      export tile data only
-maps_only       export map tilemap only
-metasprites_only export metasprite descriptors only
-source_tileset  use source tileset (image with common tiles)
-keep_duplicate_tiles do not remove duplicate tiles (default: not enabled)
-bin            export to binary format
-transposed      export transposed (column-by-column instead of row-by-row)

```

14 Todo List

Page [Coding Guidelines](#)

Update and verify this section for the modernized SDCC and toolchain

File [far_ptr.h](#)

Add link to a discussion about banking (such as, how to assign code and variables to banks)

Page [ROM/RAM Banking and MBCs](#)

Variables in RAM

Page [Using GBDK](#)

This is from GBDK 2.x docs, verify it with GBDK-2020 and modern SDCC

15 Module Index

15.1 C modules

Here is a list of all modules:

List of gbdk fonts	64
--------------------	----

16 Data Structure Index

16.1 Data Structures

Here are the data structures with brief descriptions:

__far_ptr	64
_fixed	65
atomic_flag	66
isr_nested_vector_t	66
isr_vector_t	66
joypads_t	67
metasprite_t	68
OAM_item_t	70
sfont_handle	71

17 File Index

17.1 File List

Here is a list of all files with brief descriptions:

gbdk-lib/include/assert.h	92
gbdk-lib/include/ctype.h	92
gbdk-lib/include/limits.h	232
gbdk-lib/include/rand.h	284

gbdk-lib/include/setjmp.h	286
gbdk-lib/include/stdarg.h	75
gbdk-lib/include/stdatomic.h	310
gbdk-lib/include/stdbool.h	311
gbdk-lib/include/stddef.h	311
gbdk-lib/include/stdint.h	312
gbdk-lib/include/stdio.h	318
gbdk-lib/include/stdlib.h	319
gbdk-lib/include/stdnoreturn.h	323
gbdk-lib/include/string.h	86
gbdk-lib/include/time.h	323
gbdk-lib/include/typeof.h	324
gbdk-lib/include/types.h	91
gbdk-lib/include/asm/types.h	88
gbdk-lib/include/asm/mos6502/provides.h	72
gbdk-lib/include/asm/mos6502/stdarg.h	73
gbdk-lib/include/asm/mos6502/string.h	75
gbdk-lib/include/asm/mos6502/types.h	86
gbdk-lib/include/asm/sm83/provides.h	72
gbdk-lib/include/asm/sm83/stdarg.h	74
gbdk-lib/include/asm/sm83/string.h	79
gbdk-lib/include/asm/sm83/types.h	87
gbdk-lib/include/asm/z80/provides.h	73
gbdk-lib/include/asm/z80/stdarg.h	74
gbdk-lib/include/asm/z80/string.h	82
gbdk-lib/include/asm/z80/types.h	90
gbdk-lib/include/gb/bcd.h	94
gbdk-lib/include/gb/bgb_emu.h	96
gbdk-lib/include/gb/cgb.h	96
gbdk-lib/include/gb/crash_handler.h	102
gbdk-lib/include/gb/drawing.h	102
gbdk-lib/include/gb/emu_debug.h	107

gbdk-lib/include/gb/gb.h	110
gbdk-lib/include/gb/gbdecompress.h	153
gbdk-lib/include/gb/hardware.h	157
gbdk-lib/include/gb/isr.h	202
gbdk-lib/include/gb/metasprites.h	203
gbdk-lib/include/gb/sgb.h	220
gbdk-lib/include/gbdk/bcd.h	96
gbdk-lib/include/gbdk/console.h	223
gbdk-lib/include/gbdk/emu_debug.h	107
gbdk-lib/include/gbdk/far_ptr.h	224
gbdk-lib/include/gbdk/font.h	227
gbdk-lib/include/gbdk/gbdecompress.h	156
gbdk-lib/include/gbdk/gbdk-lib.h	229
gbdk-lib/include/gbdk/incbin.h	229
gbdk-lib/include/gbdk/metasprites.h	209
gbdk-lib/include/gbdk/platform.h	230
gbdk-lib/include/gbdk/rledecompress.h	230
gbdk-lib/include/gbdk/version.h	231
gbdk-lib/include/msx/hardware.h	179
gbdk-lib/include/msx/metasprites.h	209
gbdk-lib/include/msx/msx.h	233
gbdk-lib/include/nes/hardware.h	189
gbdk-lib/include/nes/metasprites.h	211
gbdk-lib/include/nes/nes.h	256
gbdk-lib/include/nes/rgb_to_nes_macro.h	284
gbdk-lib/include/sms/gbdecompress.h	156
gbdk-lib/include/sms/hardware.h	192
gbdk-lib/include/sms/metasprites.h	217
gbdk-lib/include/sms/sms.h	287

18 Module Documentation

18.1 List of gbdk fonts

18.1.1 Description

Variables

- `uint8_t font_spect []`
- `uint8_t font_italic []`
- `uint8_t font_ibm []`
- `uint8_t font_min []`
- `uint8_t font_ibm_fixed []`

18.1.2 Variable Documentation

18.1.2.1 font_spect `uint8_t font_spect []`
The default fonts

18.1.2.2 font_italic `uint8_t font_italic []`

18.1.2.3 font_ibm `uint8_t font_ibm []`

18.1.2.4 font_min `uint8_t font_min []`

18.1.2.5 font_ibm_fixed `uint8_t font_ibm_fixed []`
Backwards compatible font

19 Data Structure Documentation

19.1 `__far_ptr` Union Reference

```
#include <gbdk-lib/include/gbdk/far_ptr.h>
```

Data Fields

- `FAR_PTR ptr`
- struct {
 `void * ofs`
 `uint16_t seg`
} `segofs`
- struct {
 `void(* fn)()`
 `uint16_t seg`
} `segfn`

19.1.1 Detailed Description

Union for working with members of a `FAR_PTR`

19.1.2 Field Documentation

19.1.2.1 ptr `FAR_PTR __far_ptr::ptr`

19.1.2.2 ofs `void* __far_ptr::ofs`

19.1.2.3 seg `uint16_t __far_ptr::seg`

19.1.2.4 segofs `struct { ... } __far_ptr::segofs`

19.1.2.5 fn `void(* __far_ptr::fn) ()`

19.1.2.6 segfn `struct { ... } __far_ptr::segfn`

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

- `gbdk-lib/include/gbdk/far_ptr.h`

19.2 `_fixed` Union Reference

```
#include <gbdk-lib/include/asm/types.h>
```

Data Fields

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

19.2.1 Detailed Description

Useful definition for working with 8 bit + 8 bit fixed point values

Use `.w` to access the variable as unsigned 16 bit type.

Use `.b.h` and `.b.l` (or just `.h` and `.l`) to directly access it's high and low unsigned 8 bit values.

19.2.2 Field Documentation

19.2.2.1 l `UBYTE _fixed::l`

19.2.2.2 h `UBYTE _fixed::h`

19.2.2.3 `"@1 struct { ... }`

19.2.2.4 `b struct { ... } _fixed::b`

19.2.2.5 `w UWORD _fixed::w`

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

- [gbdk-lib/include/asm/types.h](#)

19.3 atomic_flag Struct Reference

```
#include <gbdk-lib/include/stdatomic.h>
```

Data Fields

- unsigned char [flag](#)

19.3.1 Field Documentation

19.3.1.1 `flag unsigned char atomic_flag::flag`

The documentation for this struct was generated from the following file:

- [gbdk-lib/include/stdatomic.h](#)

19.4 isr_nested_vector_t Struct Reference

```
#include <gbdk-lib/include/gb/isr.h>
```

Data Fields

- [uint8_t](#) `opcode` [2]
- void * [func](#)

19.4.1 Field Documentation

19.4.1.1 `opcode uint8_t isr_nested_vector_t::opcode[2]`

19.4.1.2 `func void* isr_nested_vector_t::func`

The documentation for this struct was generated from the following file:

- [gbdk-lib/include/gb/isr.h](#)

19.5 isr_vector_t Struct Reference

```
#include <gbdk-lib/include/gb/isr.h>
```

Data Fields

- [uint8_t](#) `opcode`
- void * [func](#)

19.5.1 Field Documentation

19.5.1.1 opcode `uint8_t isr_vector_t::opcode`

19.5.1.2 func `void* isr_vector_t::func`

The documentation for this struct was generated from the following file:

- `gbdk-lib/include/gb/isr.h`

19.6 joypads_t Struct Reference

```
#include <gbdk-lib/include/gb/gb.h>
```

Data Fields

- `uint8_t npads`
- `union {
 struct {
 uint8_t joy0
 uint8_t joy1
 uint8_t joy2
 uint8_t joy3
 }
 uint8_t joypads [4]
};`
- `union {
 struct {
 uint8_t joy0
 uint8_t joy1
 uint8_t joy2
 uint8_t joy3
 }
 uint8_t joypads [4]
};`
- `union {
 struct {
 uint8_t joy0
 uint8_t joy1
 uint8_t joy2
 uint8_t joy3
 }
 uint8_t joypads [4]
};`
- `union {
 struct {
 uint8_t joy0
 uint8_t joy1
 uint8_t joy2
 uint8_t joy3
 }
 uint8_t joypads [4]
};`

19.6.1 Detailed Description

Multiplayer joystick structure.

Must be initialized with [joypad_init\(\)](#) first then it may be used to poll all available joypads with [joypad_ex\(\)](#)

19.6.2 Field Documentation

19.6.2.1 npads `uint8_t joypads_t::npads`

19.6.2.2 joy0 `uint8_t joypads_t::joy0`

19.6.2.3 joy1 `uint8_t joypads_t::joy1`

19.6.2.4 joy2 `uint8_t joypads_t::joy2`

19.6.2.5 joy3 `uint8_t joypads_t::joy3`

19.6.2.6 joypads `uint8_t joypads_t::joypads[4]`

19.6.2.7 "@4 `union { ... }`

19.6.2.8 "@10 `union { ... }`

19.6.2.9 "@14 `union { ... }`

19.6.2.10 "@18 `union { ... }`

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

- `gbdk-lib/include/gb/gb.h`
- `gbdk-lib/include/msx/msx.h`
- `gbdk-lib/include/nes/nes.h`
- `gbdk-lib/include/sms/sms.h`

19.7 metasprite_t Struct Reference

```
#include <gbdk-lib/include/gb/metasprites.h>
```

Data Fields

- `int8_t dy`
- `int8_t dx`
- `uint8_t dtile`
- `uint8_t props`

19.7.1 Detailed Description

Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(uint8_t) Start tile relative to the metasprites own set of tiles
<i>props</i>	(uint8_t) Property Flags

Metasprites are built from multiple [metasprite_t](#) items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of [metasprite_t](#) items (which may vary based on how many sprites are required for that particular frame).

A metasprite frame is terminated with a {metasprite_end} entry.

Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(uint8_t) Start tile relative to the metasprites own set of tiles

Metasprites are built from multiple [metasprite_t](#) items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of [metasprite_t](#) items (which may vary based on how many sprites are required for that particular frame).

A metasprite frame is terminated with a {metasprite_end} entry.

19.7.2 Field Documentation

19.7.2.1 dy [int8_t](#) metasprite_t::dy

19.7.2.2 dx [int8_t](#) metasprite_t::dx

19.7.2.3 dtile [uint8_t](#) metasprite_t::dtile

19.7.2.4 props [uint8_t](#) metasprite_t::props

The documentation for this struct was generated from the following file:

- [gbdk-lib/include/gb/metasprites.h](#)

19.8 OAM_item_t Struct Reference

```
#include <gbdk-lib/include/gb/gb.h>
```

Data Fields

- [uint8_t y](#)
- [uint8_t x](#)
- [uint8_t tile](#)
- [uint8_t prop](#)

19.8.1 Detailed Description

Sprite Attributes structure

Parameters

<i>x</i>	X Coordinate of the sprite on screen
<i>y</i>	Y Coordinate of the sprite on screen
<i>tile</i>	Sprite tile number (see set_sprite_tile)
<i>prop</i>	OAM Property Flags (see set_sprite_prop)

Sprite Attributes structure

Parameters

<i>x</i>	X Coordinate of the sprite on screen
<i>y</i>	Y Coordinate of the sprite on screen - 1
<i>tile</i>	Sprite tile number (see set_sprite_tile)
<i>prop</i>	OAM Property Flags (see set_sprite_prop)

19.8.2 Field Documentation

19.8.2.1 `y` [uint8_t](#) `OAM_item_t::y`

19.8.2.2 `x` [uint8_t](#) `OAM_item_t::x`

19.8.2.3 `tile` [uint8_t](#) `OAM_item_t::tile`

19.8.2.4 `prop` [uint8_t](#) `OAM_item_t::prop`

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

- `gbdk-lib/include/gb/gb.h`
- `gbdk-lib/include/msx/msx.h`
- `gbdk-lib/include/nes/nes.h`

19.9 `sfont_handle` Struct Reference

```
#include <gbdk-lib/include/gbdk/font.h>
```

Data Fields

- [uint8_t](#) `first_tile`
- `void *` `font`

19.9.1 Detailed Description

Font handle structure

19.9.2 Field Documentation

19.9.2.1 `first_tile` [uint8_t](#) `sfont_handle::first_tile`

First tile used for font

19.9.2.2 font `void* sfont_handle::font`

Pointer to the base of the font

The documentation for this struct was generated from the following file:

- `gbdk-lib/include/gbdk/font.h`

20 File Documentation

20.1 `docs/pages/01_getting_started.md` File Reference

20.2 `docs/pages/02_links_and_tools.md` File Reference

20.3 `docs/pages/03_using_gbdk.md` File Reference

20.4 `docs/pages/04_coding_guidelines.md` File Reference

20.5 `docs/pages/05_banking_mbc5.md` File Reference

20.6 `docs/pages/06_toolchain.md` File Reference

20.7 `docs/pages/06b_supported_consoles.md` File Reference

20.8 `docs/pages/07_sample_programs.md` File Reference

20.9 `docs/pages/08_faq.md` File Reference

20.10 `docs/pages/09_migrating_new_versions.md` File Reference

20.11 `docs/pages/10_release_notes.md` File Reference

20.12 `docs/pages/20_toolchain_settings.md` File Reference

20.13 `docs/pages/docs_index.md` File Reference

20.14 `gbdk-lib/include/asm/mos6502/provides.h` File Reference

Macros

- `#define USE_C_MEMCPY 0`
- `#define USE_C_STRCPY 0`
- `#define USE_C_STRCMP 1`

20.14.1 Macro Definition Documentation

20.14.1.1 `USE_C_MEMCPY` `#define USE_C_MEMCPY 0`

20.14.1.2 `USE_C_STRCPY` `#define USE_C_STRCPY 0`

20.14.1.3 `USE_C_STRCMP` `#define USE_C_STRCMP 1`

20.15 `gbdk-lib/include/asm/sm83/provides.h` File Reference

Macros

- `#define USE_C_MEMCPY 0`
- `#define USE_C_STRCPY 0`
- `#define USE_C_STRCMP 0`

20.15.1 Macro Definition Documentation

20.15.1.1 USE_C_MEMCPY `#define USE_C_MEMCPY 0`

20.15.1.2 USE_C_STRCPY `#define USE_C_STRCPY 0`

20.15.1.3 USE_C_STRCMP `#define USE_C_STRCMP 0`

20.16 gbdk-lib/include/asm/z80/provides.h File Reference

Macros

- `#define USE_C_MEMCPY 0`
- `#define USE_C_STRCPY 0`
- `#define USE_C_STRCMP 1`

20.16.1 Macro Definition Documentation

20.16.1.1 USE_C_MEMCPY `#define USE_C_MEMCPY 0`

20.16.1.2 USE_C_STRCPY `#define USE_C_STRCPY 0`

20.16.1.3 USE_C_STRCMP `#define USE_C_STRCMP 1`

20.17 gbdk-lib/include/asm/mos6502/stdarg.h File Reference

Macros

- `#define va_start(list, last) list = (unsigned char *)&last + sizeof(last)`
- `#define va_arg(list, type) *((type *)((list += sizeof(type)) - sizeof(type)))`
- `#define va_end(list)`

Typedefs

- `typedef unsigned char * va_list`

20.17.1 Macro Definition Documentation

20.17.1.1 va_start `#define va_start(
list,
last) list = (unsigned char *)&last + sizeof(last)`

20.17.1.2 va_arg `#define va_arg(
list,
type) *((type *)((list += sizeof(type)) - sizeof(type)))`

20.17.1.3 va_end `#define va_end(
list)`

20.17.2 Typedef Documentation

20.17.2.1 va_list `typedef unsigned char* va_list`

20.18 gbdk-lib/include/asm/sm83/stdarg.h File Reference

Macros

- `#define va_start(list, last) list = (unsigned char *)&last + sizeof(last)`
- `#define va_arg(list, type) *((type *)((list += sizeof(type)) - sizeof(type)))`
- `#define va_end(list)`

Typedefs

- `typedef unsigned char * va_list`

20.18.1 Macro Definition Documentation

20.18.1.1 va_start `#define va_start(
list,
last) list = (unsigned char *)&last + sizeof(last)`

20.18.1.2 va_arg `#define va_arg(
list,
type) *((type *)((list += sizeof(type)) - sizeof(type)))`

20.18.1.3 va_end `#define va_end(
list)`

20.18.2 Typedef Documentation

20.18.2.1 va_list `typedef unsigned char* va_list`

20.19 gbdk-lib/include/asm/z80/stdarg.h File Reference

Macros

- `#define va_start(list, last) list = (unsigned char *)&last + sizeof(last)`
- `#define va_arg(list, type) *((type *)((list += sizeof(type)) - sizeof(type)))`
- `#define va_end(list)`

Typedefs

- `typedef unsigned char * va_list`

20.19.1 Macro Definition Documentation

```

20.19.1.1 va_start #define va_start(
    list,
    last ) list = (unsigned char *)&last + sizeof(last)

```

```

20.19.1.2 va_arg #define va_arg(
    list,
    type ) *((type *) ((list += sizeof(type)) - sizeof(type)))

```

```

20.19.1.3 va_end #define va_end(
    list )

```

20.19.2 Typedef Documentation

```

20.19.2.1 va_list typedef unsigned char* va_list

```

20.20 gbdk-lib/include/stdarg.h File Reference

```
#include <asm/sm83/stdarg.h>
```

20.21 gbdk-lib/include/asm/mos6502/string.h File Reference

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

Functions

- char * [strcpy](#) (char *dest, const char *src) [OLDCALL](#)
- int [strcmp](#) (const char *s1, const char *s2)
- void * [memcpy](#) (void *dest, const void *src, [size_t](#) len)
- void * [memmove](#) (void *dest, const void *src, [size_t](#) n) [OLDCALL](#)
- void * [memset](#) (void *s, int c, [size_t](#) n)
- char * [reverse](#) (char *s) [NONBANKED](#)
- char * [strcat](#) (char *s1, const char *s2) [NONBANKED](#)
- int [strlen](#) (const char *s) [OLDCALL](#)
- char * [strncat](#) (char *s1, const char *s2, int n) [NONBANKED](#)
- int [strncmp](#) (const char *s1, const char *s2, int n) [NONBANKED](#)
- char * [strncpy](#) (char *s1, const char *s2, int n) [NONBANKED](#)
- int [memcmp](#) (const void *buf1, const void *buf2, [size_t](#) count)

20.21.1 Detailed Description

Generic string functions.

20.21.2 Function Documentation

```

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

```

Copies the string pointed to by **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

<i>dest</i>	Array to copy into
<i>src</i>	Array to copy from

Returns

A pointer to *dest*

20.21.2.2 strcmp() `int strcmp (`
 `const char * s1,`
 `const char * s2)`

Compares strings

Parameters

<i>s1</i>	First string to compare
<i>s2</i>	Second string to compare

Returns:

- > 0 if **s1** $>$ **s2**
- 0 if **s1** $==$ **s2**
- < 0 if **s1** $<$ **s2**

20.21.2.3 memcpy() `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

<i>dest</i>	Buffer to copy into
<i>src</i>	Buffer to copy from
<i>len</i>	Number of Bytes to copy

20.21.2.4 memmove() `void* memmove (`
 `void * dest,`
 `const void * src,`
 `size_t n)`

Copies *n* bytes from memory area *src* to memory area *dest*, areas may overlap

20.21.2.5 memset() `void* memset (`
 `void * s,`
 `int c,`
 `size_t n)`

Fills the memory region **s** with **n** bytes using value **c**

Parameters

<i>s</i>	Buffer to fill
<i>c</i>	char value to fill with (truncated from int)
<i>n</i>	Number of bytes to fill

20.21.2.6 reverse() `char* reverse (`
 `char * s)`

Reverses the characters in a string

Parameters

<i>s</i>	Pointer to string to reverse.
----------	-------------------------------

For example 'abcdefg' will become 'gfedcba'.

Banked as the string must be modifiable.

Returns: Pointer to **s**

20.21.2.7 strcat() `char* strcat (`
 `char * s1,`
 `const char * s2)`

Concatenate Strings. Appends string **s2** to the end of string **s1**

Parameters

<i>s1</i>	String to append onto
<i>s2</i>	String to copy from

For example 'abc' and 'def' will become 'abcdef'.

String **s1** must be large enough to store both **s1** and **s2**.

Returns: Pointer to **s1**

20.21.2.8 strlen() `int strlen (`
 `const char * s)`

Calculates the length of a string

Parameters

<i>s</i>	String to calculate length of
----------	-------------------------------

Returns: Length of string not including the terminating '\0' character.

20.21.2.9 strncat() `char* strncat (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Concatenate at most **n** characters from string **s2** onto the end of **s1**.

Parameters

<i>s1</i>	String to append onto
<i>s2</i>	String to copy from
<i>n</i>	Max number of characters to copy from s2

String **s1** must be large enough to store both **s1** and **n** characters of **s2**
Returns: Pointer to **s1**

20.21.2.10 strncmp() `int strncmp (`
 `const char * s1,`
 `const char * s2,`
 `int n)`

Compare strings (at most **n** characters):

Parameters

<i>s1</i>	First string to compare
<i>s2</i>	Second string to compare
<i>n</i>	Max number of characters to compare

Returns:

- > 0 if **s1** $>$ **s2**
- 0 if **s1** $==$ **s2**
- < 0 if **s1** $<$ **s2**

20.21.2.11 strncpy() `char* strncpy (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Copy **n** characters from string **s2** to **s1**

Parameters

<i>s1</i>	String to copy into
<i>s2</i>	String to copy from
<i>n</i>	Max number of characters to copy from s2

If **s2** is shorter than **n**, the remaining bytes in **s1** are filled with `\0`.

Warning: If there is no `\0` in the first **n** bytes of **s2** then **s1** will not be null terminated.

Returns: Pointer to **s1**

20.21.2.12 memcmp() `int memcmp (`
 `const void * buf1,`
 `const void * buf2,`
 `size_t count)`

Compares buffers

Parameters

<i>buf1</i>	First buffer to compare
<i>buf2</i>	Second buffer to compare
<i>count</i>	Buffer length

Returns:

- > 0 if **buf1** $>$ **buf2**
- 0 if **buf1** $==$ **buf2**

- < 0 if **buf1** < **buf2**

20.22 gbdk-lib/include/asm/sm83/string.h File Reference

#include <types.h>

Functions

- char * [strcpy](#) (char *dest, const char *src) [OLDCALL PRESERVES_REGS\(b](#)
- int [strcmp](#) (const char *s1, const char *s2) [OLDCALL PRESERVES_REGS\(b](#)
- void * [memcpy](#) (void *dest, const void *src, [size_t](#) len)
- void * [memmove](#) (void *dest, const void *src, [size_t](#) n)
- void * [memset](#) (void *s, int c, [size_t](#) n) [OLDCALL PRESERVES_REGS\(b](#)
- char * [reverse](#) (char *s) [OLDCALL PRESERVES_REGS\(b](#)
- char * [strcat](#) (char *s1, const char *s2)
- int [strlen](#) (const char *s) [OLDCALL PRESERVES_REGS\(b](#)
- 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)
- int [memcmp](#) (const void *buf1, const void *buf2, [size_t](#) count) [OLDCALL](#)

Variables

- char [c](#)

20.22.1 Detailed Description

Generic string functions.

20.22.2 Function Documentation

20.22.2.1 strcpy() char* strcpy (

char * dest,

const char * src)

Copies the string pointed to by **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

<i>dest</i>	Array to copy into
<i>src</i>	Array to copy from

Returns

A pointer to dest

20.22.2.2 strcmp() int strcmp (

const char * s1,

const char * s2)

Compares strings

Parameters

<i>s1</i>	First string to compare
<i>s2</i>	Second string to compare

Returns:

- > 0 if **s1** $>$ **s2**
- 0 if **s1** $==$ **s2**
- < 0 if **s1** $<$ **s2**

20.22.2.3 memcpy() `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

<i>dest</i>	Buffer to copy into
<i>src</i>	Buffer to copy from
<i>len</i>	Number of Bytes to copy

20.22.2.4 memmove() `void* memmove (`
 `void * dest,`
 `const void * src,`
 `size_t n)`

Copies *n* bytes from memory area *src* to memory area *dest*, areas may overlap

20.22.2.5 memset() `void* memset (`
 `void * s,`
 `int c,`
 `size_t n)`

Fills the memory region **s** with **n** bytes using value **c**

Parameters

<i>s</i>	Buffer to fill
<i>c</i>	char value to fill with (truncated from int)
<i>n</i>	Number of bytes to fill

20.22.2.6 reverse() `char* reverse (`
 `char * s)`

Reverses the characters in a string

Parameters

<i>s</i>	Pointer to string to reverse.
----------	-------------------------------

For example 'abcdefg' will become 'gfedcba'.

Banked as the string must be modifiable.

Returns: Pointer to **s**

20.22.2.7 strcat() `char* strcat (`
 `char * s1,`
 `const char * s2)`

Concatenate Strings. Appends string **s2** to the end of string **s1**

Parameters

s1	String to append onto
s2	String to copy from

For example 'abc' and 'def' will become 'abcdef'.

String **s1** must be large enough to store both **s1** and **s2**.

Returns: Pointer to **s1**

20.22.2.8 strlen() `int strlen (`
 `const char * s)`

Calculates the length of a string

Parameters

s	String to calculate length of
----------	-------------------------------

Returns: Length of string not including the terminating '\0' character.

20.22.2.9 strncat() `char* strncat (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Concatenate at most **n** characters from string **s2** onto the end of **s1**.

Parameters

s1	String to append onto
s2	String to copy from
n	Max number of characters to copy from s2

String **s1** must be large enough to store both **s1** and **n** characters of **s2**

Returns: Pointer to **s1**

20.22.2.10 strncmp() `int strncmp (`
 `const char * s1,`
 `const char * s2,`
 `int n)`

Compare strings (at most **n** characters):

Parameters

s1	First string to compare
s2	Second string to compare
n	Max number of characters to compare

Returns zero if the strings are identical, or non-zero if they are not (see below).
Returns:

- > 0 if **s1** $>$ **s2** (at first non-matching byte)
- 0 if **s1** $==$ **s2**
- < 0 if **s1** $<$ **s2** (at first non-matching byte)

20.22.2.11 strncpy() `char* strncpy (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Copy **n** characters from string **s2** to **s1**

Parameters

<i>s1</i>	String to copy into
<i>s2</i>	String to copy from
<i>n</i>	Max number of characters to copy from s2

If **s2** is shorter than **n**, the remaining bytes in **s1** are filled with `\0`.
Warning: If there is no `\0` in the first **n** bytes of **s2** then **s1** will not be null terminated.
Returns: Pointer to **s1**

20.22.2.12 memcmp() `int memcmp (`
 `const void * buf1,`
 `const void * buf2,`
 `size_t count)`

Compare up to **count** bytes in buffers **buf1** and **buf2**

Parameters

<i>buf1</i>	Pointer to First buffer to compare
<i>buf2</i>	Pointer to Second buffer to compare
<i>count</i>	Max number of bytes to compare

Returns zero if the buffers are identical, or non-zero if they are not (see below).
Returns:

- > 0 if **buf1** $>$ **buf2** (at first non-matching byte)
- 0 if **buf1** $==$ **buf2**
- < 0 if **buf1** $<$ **buf2** (at first non-matching byte)

20.22.3 Variable Documentation

20.22.3.1 c `void c`

20.23 gbdk-lib/include/asm/z80/string.h File Reference

`#include <types.h>`

Functions

- char * [strcpy](#) (char *dest, const char *src) [OLDCALL](#)
- int [strcmp](#) (const char *s1, const char *s2)
- void * [memcpy](#) (void *dest, const void *src, [size_t](#) len)
- void * [memmove](#) (void *dest, const void *src, [size_t](#) n) [OLDCALL](#)
- void * [memset](#) (void *s, int c, [size_t](#) n) [Z88DK_CALLEE](#)
- char * [reverse](#) (char *s) [NONBANKED](#)
- char * [strcat](#) (char *s1, const char *s2) [NONBANKED](#)
- int [strlen](#) (const char *s) [OLDCALL](#)
- char * [strncat](#) (char *s1, const char *s2, int n) [NONBANKED](#)
- int [strncmp](#) (const char *s1, const char *s2, int n) [NONBANKED](#)
- char * [strncpy](#) (char *s1, const char *s2, int n) [NONBANKED](#)
- int [memcmp](#) (const void *buf1, const void *buf2, [size_t](#) count) [Z88DK_CALLEE](#)

20.23.1 Detailed Description

Generic string functions.

20.23.2 Function Documentation

20.23.2.1 strcpy() char* strcpy (

 char * dest,

 const char * src)

Copies the string pointed to by **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

<i>dest</i>	Array to copy into
<i>src</i>	Array to copy from

Returns

A pointer to dest

20.23.2.2 strcmp() int strcmp (

 const char * s1,

 const char * s2)

Compares strings

Parameters

<i>s1</i>	First string to compare
<i>s2</i>	Second string to compare

Returns:

- > 0 if **s1** > **s2**
- 0 if **s1** == **s2**
- < 0 if **s1** < **s2**

20.23.2.3 memcpy() `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

<i>dest</i>	Buffer to copy into
<i>src</i>	Buffer to copy from
<i>len</i>	Number of Bytes to copy

20.23.2.4 memmove() `void* memmove (`
 `void * dest,`
 `const void * src,`
 `size_t n)`

Copies `n` bytes from memory area `src` to memory area `dest`, areas may overlap

20.23.2.5 memset() `void* memset (`
 `void * s,`
 `int c,`
 `size_t n)`

Fills the memory region `s` with `n` bytes using value `c`

Parameters

<i>s</i>	Buffer to fill
<i>c</i>	char value to fill with (truncated from int)
<i>n</i>	Number of bytes to fill

20.23.2.6 reverse() `char* reverse (`
 `char * s)`

Reverses the characters in a string

Parameters

<i>s</i>	Pointer to string to reverse.
----------	-------------------------------

For example 'abcdefg' will become 'gfedcba'.
Banked as the string must be modifiable.
Returns: Pointer to `s`

20.23.2.7 strcat() `char* strcat (`
 `char * s1,`
 `const char * s2)`

Concatenate Strings. Appends string `s2` to the end of string `s1`

Parameters

<i>s1</i>	String to append onto
<i>s2</i>	String to copy from

For example 'abc' and 'def' will become 'abcdef'.
 String **s1** must be large enough to store both **s1** and **s2**.
 Returns: Pointer to **s1**

20.23.2.8 strlen() `int strlen (`
 `const char * s)`

Calculates the length of a string

Parameters

<i>s</i>	String to calculate length of
----------	-------------------------------

Returns: Length of string not including the terminating '\0' character.

20.23.2.9 strncat() `char* strncat (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Concatenate at most **n** characters from string **s2** onto the end of **s1**.

Parameters

<i>s1</i>	String to append onto
<i>s2</i>	String to copy from
<i>n</i>	Max number of characters to copy from s2

String **s1** must be large enough to store both **s1** and **n** characters of **s2**
 Returns: Pointer to **s1**

20.23.2.10 strncmp() `int strncmp (`
 `const char * s1,`
 `const char * s2,`
 `int n)`

Compare strings (at most **n** characters):

Parameters

<i>s1</i>	First string to compare
<i>s2</i>	Second string to compare
<i>n</i>	Max number of characters to compare

Returns:

- **> 0** if **s1 > s2**
- **0** if **s1 == s2**
- **< 0** if **s1 < s2**

20.23.2.11 strncpy() `char* strncpy (`
 `char * s1,`
 `const char * s2,`
 `int n)`

Copy **n** characters from string **s2** to **s1**

Parameters

<i>s1</i>	String to copy into
<i>s2</i>	String to copy from
<i>n</i>	Max number of characters to copy from s2

If **s2** is shorter than **n**, the remaining bytes in **s1** are filled with `\0`.

Warning: If there is no `\0` in the first **n** bytes of **s2** then **s1** will not be null terminated.

Returns: Pointer to **s1**

20.23.2.12 memcmp() `int memcmp (`
 `const void * buf1,`
 `const void * buf2,`
 `size_t count)`

Compares buffers

Parameters

<i>buf1</i>	First buffer to compare
<i>buf2</i>	Second buffer to compare
<i>count</i>	Buffer length

Returns:

- `> 0` if **buf1** `>` **buf2**
- `0` if **buf1** `==` **buf2**
- `< 0` if **buf1** `<` **buf2**

20.24 gbdk-lib/include/string.h File Reference

```
#include <asm/sm83/string.h>
```

20.24.1 Detailed Description

Generic string functions.

20.25 gbdk-lib/include/asm/mos6502/types.h File Reference

Macros

- `#define __SIZE_T_DEFINED`

Typedefs

- typedef signed char [INT8](#)
- typedef unsigned char [UINT8](#)
- typedef signed int [INT16](#)
- typedef unsigned int [UINT16](#)
- typedef signed long [INT32](#)
- typedef unsigned long [UINT32](#)
- typedef unsigned int [size_t](#)
- typedef unsigned int [clock_t](#)

20.25.1 Detailed Description

Types definitions for the gb.

20.25.2 Macro Definition Documentation

20.25.2.1 `__SIZE_T_DEFINED` `#define __SIZE_T_DEFINED`

20.25.3 Typedef Documentation

20.25.3.1 `INT8` `typedef signed char INT8`
Signed eight bit.

20.25.3.2 `UINT8` `typedef unsigned char UINT8`
Unsigned eight bit.

20.25.3.3 `INT16` `typedef signed int INT16`
Signed sixteen bit.

20.25.3.4 `UINT16` `typedef unsigned int UINT16`
Unsigned sixteen bit.

20.25.3.5 `INT32` `typedef signed long INT32`
Signed 32 bit.

20.25.3.6 `UINT32` `typedef unsigned long UINT32`
Unsigned 32 bit.

20.25.3.7 `size_t` `typedef unsigned int size_t`

20.25.3.8 `clock_t` `typedef unsigned int clock_t`
Returned from clock

See also

[clock](#)

20.26 gbdk-lib/include/asm/sm83/types.h File Reference

Macros

- `#define __SIZE_T_DEFINED`

Typedefs

- `typedef signed char INT8`
- `typedef unsigned char UINT8`
- `typedef signed int INT16`
- `typedef unsigned int UINT16`
- `typedef signed long INT32`
- `typedef unsigned long UINT32`
- `typedef unsigned int size_t`
- `typedef unsigned int clock_t`

20.26.1 Detailed Description

Types definitions for the gb.

20.26.2 Macro Definition Documentation

20.26.2.1 `__SIZE_T_DEFINED` `#define __SIZE_T_DEFINED`

20.26.3 Typedef Documentation

20.26.3.1 `INT8` `typedef signed char INT8`
Signed eight bit.

20.26.3.2 `UINT8` `typedef unsigned char UINT8`
Unsigned eight bit.

20.26.3.3 `INT16` `typedef signed int INT16`
Signed sixteen bit.

20.26.3.4 `UINT16` `typedef unsigned int UINT16`
Unsigned sixteen bit.

20.26.3.5 `INT32` `typedef signed long INT32`
Signed 32 bit.

20.26.3.6 `UINT32` `typedef unsigned long UINT32`
Unsigned 32 bit.

20.26.3.7 `size_t` `typedef unsigned int size_t`

20.26.3.8 `clock_t` `typedef unsigned int clock_t`
Returned from clock

See also

[clock](#)

20.27 gbdk-lib/include/asm/types.h File Reference

`#include <asm/sm83/types.h>`

Data Structures

- [union _fixed](#)

Macros

- `#define` [OLDCALL](#)
- `#define` [PRESERVES_REGS\(...\)](#)
- `#define` [NAKED](#)
- `#define` [SFR](#)
- `#define` [AT\(A\)](#)
- `#define` [NONBANKED](#)
- `#define` [BANKED](#)
- `#define` [CRITICAL](#)
- `#define` [INTERRUPT](#)

Typedefs

- typedef [INT8](#) [BOOLEAN](#)
- typedef [INT8](#) [BYTE](#)
- typedef [UINT8](#) [UBYTE](#)
- typedef [INT16](#) [WORD](#)
- typedef [UINT16](#) [UWORD](#)
- typedef [INT32](#) [LWORD](#)
- typedef [UINT32](#) [ULWORD](#)
- typedef [INT32](#) [DWORD](#)
- typedef [UINT32](#) [UDWORD](#)
- typedef union [_fixed](#) [fixed](#)

20.27.1 Detailed Description

Shared types definitions.

20.27.2 Macro Definition Documentation

20.27.2.1 OLDCALL `#define OLDCALL`

20.27.2.2 PRESERVES_REGS `#define PRESERVES_REGS(
...)`

20.27.2.3 NAKED `#define NAKED`

20.27.2.4 SFR `#define SFR`

20.27.2.5 AT `#define AT(
A)`

20.27.2.6 NONBANKED `#define NONBANKED`

20.27.2.7 BANKED `#define BANKED`

20.27.2.8 CRITICAL `#define CRITICAL`

20.27.2.9 INTERRUPT `#define INTERRUPT`

20.27.3 Typedef Documentation

20.27.3.1 BOOLEAN typedef [INT8](#) [BOOLEAN](#)
TRUE or FALSE.

20.27.3.2 BYTE `typedef INT8 BYTE`
Signed 8 bit.

20.27.3.3 UBYTE `typedef UINT8 UBYTE`
Unsigned 8 bit.

20.27.3.4 WORD `typedef INT16 WORD`
Signed 16 bit

20.27.3.5 UWORD `typedef UINT16 UWORD`
Unsigned 16 bit

20.27.3.6 LWORD `typedef INT32 LWORD`
Signed 32 bit

20.27.3.7 ULWORD `typedef UINT32 ULWORD`
Unsigned 32 bit

20.27.3.8 DWORD `typedef INT32 DWORD`
Signed 32 bit

20.27.3.9 UDWORD `typedef UINT32 UDWORD`
Unsigned 32 bit

20.27.3.10 fixed `typedef union _fixed fixed`
Useful definition for working with 8 bit + 8 bit fixed point values
Use `.w` to access the variable as unsigned 16 bit type.
Use `.b.h` and `.b.l` (or just `.h` and `.l`) to directly access it's high and low unsigned 8 bit values.

20.28 gbdk-lib/include/asm/z80/types.h File Reference

Macros

- `#define Z88DK_CALLEE`
- `#define Z88DK_FASTCALL`
- `#define __SIZE_T_DEFINED`

Typedefs

- `typedef signed char INT8`
- `typedef unsigned char UINT8`
- `typedef signed int INT16`
- `typedef unsigned int UINT16`
- `typedef signed long INT32`
- `typedef unsigned long UINT32`
- `typedef unsigned int size_t`
- `typedef unsigned int clock_t`

20.28.1 Detailed Description

Types definitions for the gb.

20.28.2 Macro Definition Documentation

20.28.2.1 Z88DK_CALLEE `#define Z88DK_CALLEE`

20.28.2.2 Z88DK_FASTCALL `#define Z88DK_FASTCALL`

20.28.2.3 __SIZE_T_DEFINED `#define __SIZE_T_DEFINED`

20.28.3 Typedef Documentation

20.28.3.1 INT8 typedef signed char [INT8](#)
Signed eight bit.

20.28.3.2 UINT8 typedef unsigned char [UINT8](#)
Unsigned eight bit.

20.28.3.3 INT16 typedef signed int [INT16](#)
Signed sixteen bit.

20.28.3.4 UINT16 typedef unsigned int [UINT16](#)
Unsigned sixteen bit.

20.28.3.5 INT32 typedef signed long [INT32](#)
Signed 32 bit.

20.28.3.6 UINT32 typedef unsigned long [UINT32](#)
Unsigned 32 bit.

20.28.3.7 size_t typedef unsigned int [size_t](#)

20.28.3.8 clock_t typedef unsigned int [clock_t](#)
Returned from clock

See also

[clock](#)

20.29 gbdk-lib/include/types.h File Reference

`#include <asm/types.h>`

Macros

- `#define NULL 0`
- `#define FALSE 0`
- `#define TRUE 1`

Typedefs

- typedef void * [POINTER](#)

20.29.1 Detailed Description

Basic types.
Directly include the port specific file.

20.29.2 Macro Definition Documentation

20.29.2.1 NULL `#define NULL 0`
Good 'ol NULL.

20.29.2.2 FALSE `#define FALSE 0`
A 'false' value.

20.29.2.3 TRUE `#define TRUE 1`
A 'true' value.

20.29.3 Typedef Documentation

20.29.3.1 POINTER `typedef void* POINTER`
No longer used.

20.30 gbdk-lib/include/assert.h File Reference

Macros

- `#define assert(x) ((x) ? (void)0 : __assert(#x, __func__, __FILE__, __LINE__))`

Functions

- `void __assert (const char *expression, const char *functionname, const char *filename, unsigned int linenumber)`

20.30.1 Macro Definition Documentation

20.30.1.1 assert `#define assert(`
 `x) ((x) ? (void)0 : __assert(#x, __func__, __FILE__, __LINE__))`

20.30.2 Function Documentation

20.30.2.1 __assert() `void __assert (`
 `const char * expression,`
 `const char * functionname,`
 `const char * filename,`
 `unsigned int linenumber)`

20.31 gbdk-lib/include/ctype.h File Reference

```
#include <types.h>
#include <stdbool.h>
```

Functions

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

20.31.1 Detailed Description

Character type functions.

20.31.2 Function Documentation

20.31.2.1 isalpha() `bool isalpha (`
 char `c`)

Returns TRUE if the character `c` is a letter (a-z, A-Z), otherwise FALSE

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.2 isupper() `bool isupper (`
 char `c`)

Returns TRUE if the character `c` is an uppercase letter (A-Z), otherwise FALSE

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.3 islower() `bool islower (`
 char `c`)

Returns TRUE if the character `c` is a lowercase letter (a-z), otherwise FALSE

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.4 isdigit() `bool isdigit (`
 char `c`)

Returns TRUE if the character `c` is a digit (0-9), otherwise FALSE

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.5 isspace() `bool isspace (`
 `char c)`

Returns TRUE if the character **c** is a space (' '), tab (\t), or newline (\n) character, otherwise FALSE

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.6 toupper() `char toupper (`
 `char c)`

Returns uppercase version of character **c** if it is a letter (a-z), otherwise it returns the input value unchanged.

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.31.2.7 tolower() `char tolower (`
 `char c)`

Returns lowercase version of character **c** if it is a letter (A-Z), otherwise it returns the input value unchanged.

Parameters

<code>c</code>	Character to test
----------------	-------------------

20.32 gbdk-lib/include/gb/bcd.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define BCD_HEX(v) ((BCD)(v))`
- `#define MAKE_BCD(v) BCD_HEX(0x ## v)`

Typedefs

- `typedef uint32_t BCD`

Functions

- `void uint2bcd (uint16_t i, BCD *value) OLDCALL`
- `void bcd_add (BCD *sour, const BCD *value) OLDCALL`
- `void bcd_sub (BCD *sour, const BCD *value) OLDCALL`
- `uint8_t bcd2text (const BCD *bcd, uint8_t tile_offset, uint8_t *buffer) OLDCALL`

20.32.1 Detailed Description

Support for working with BCD (Binary Coded Decimal)
See the example BCD project for additional details.

20.32.2 Macro Definition Documentation

20.32.2.1 BCD_HEX `#define BCD_HEX(
 v) ((BCD)(v))`

20.32.2.2 MAKE_BCD `#define MAKE_BCD(
 v) BCD_HEX(0x ## v)`

Converts an integer value into BCD format

A maximum of 8 digits may be used

20.32.3 Typedef Documentation

20.32.3.1 BCD `typedef uint32_t BCD`

20.32.4 Function Documentation

20.32.4.1 uint2bcd() `void uint2bcd (
 uint16_t i,
 BCD * value)`

Converts integer *i* into BCD format (Binary Coded Decimal)

Parameters

<i>i</i>	Numeric value to convert
<i>value</i>	Pointer to a BCD variable to store the converted result

20.32.4.2 bcd_add() `void bcd_add (
 BCD * sour,
 const BCD * value)`

Adds two numbers in BCD format: **sour += value**

Parameters

<i>sour</i>	Pointer to a BCD value to add to (and where the result is stored)
<i>value</i>	Pointer to the BCD value to add to sour

20.32.4.3 bcd_sub() `void bcd_sub (
 BCD * sour,
 const BCD * value)`

Subtracts two numbers in BCD format: **sour -= value**

Parameters

<i>sour</i>	Pointer to a BCD value to subtract from (and where the result is stored)
<i>value</i>	Pointer to the BCD value to subtract from sour

20.32.4.4 bcd2text() `uint8_t bcd2text (`
 `const BCD * bcd,`
 `uint8_t tile_offset,`
 `uint8_t * buffer)`

Convert a BCD number into an asciiz (null terminated) string and return the length

Parameters

<i>bcd</i>	Pointer to BCD value to convert
<i>tile_offset</i>	Optional per-character offset value to add (use 0 for none)
<i>buffer</i>	Buffer to store the result in

Returns: Length in characters (always 8)

buffer should be large enough to store the converted string (9 bytes: 8 characters + 1 for terminator)

There are a couple different ways to use **tile_offset**. For example:

- It can be the Index of the Font Tile '0' in VRAM to allow the buffer to be used directly with [set_bkg_tiles](#).
- It can also be set to the ascii value for character '0' so that the buffer is a normal string that can be passed to [printf](#).

20.33 gbdk-lib/include/gbdk/bcd.h File Reference

```
#include <gb/bcd.h>
```

20.34 gbdk-lib/include/gb/bgb_emu.h File Reference

```
#include <gbdk/emu_debug.h>
```

20.34.1 Detailed Description

Shim for legacy use of [bgb_emu.h](#) which has been migrated to [emu_debug.h](#)

See the [emu_debug](#) example project included with gbdk.

20.35 gbdk-lib/include/gb/cgb.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define RGB(r, g, b) (((uint16_t)((b) & 0x1f) << 10) | (((g) & 0x1f) << 5) | (((r) & 0x1f) << 0))`
- `#define RGB8(r, g, b) (((uint16_t)((r) >> 3) & 0x1f) | (((g) >> 3) & 0x1f) << 5) | (((b) >> 3) & 0x1f) << 10))`
- `#define RGBHTML(RGB24bit) (RGB8((((RGB24bit) >> 16) & 0xff), (((RGB24bit) >> 8) & 0xff), ((RGB24bit) & 0xff)))`
- `#define RGB_RED RGB(31, 0, 0)`
- `#define RGB_DARKRED RGB(15, 0, 0)`
- `#define RGB_GREEN RGB(0, 31, 0)`
- `#define RGB_DARKGREEN RGB(0, 15, 0)`
- `#define RGB_BLUE RGB(0, 0, 31)`
- `#define RGB_DARKBLUE RGB(0, 0, 15)`
- `#define RGB_YELLOW RGB(31, 31, 0)`

- `#define RGB_DARKYELLOW RGB(21, 21, 0)`
- `#define RGB_CYAN RGB(0, 31, 31)`
- `#define RGB_AQUA RGB(28, 5, 22)`
- `#define RGB_PINK RGB(31, 0, 31)`
- `#define RGB_PURPLE RGB(21, 0, 21)`
- `#define RGB_BLACK RGB(0, 0, 0)`
- `#define RGB_DARKGRAY RGB(10, 10, 10)`
- `#define RGB_LIGHTGRAY RGB(21, 21, 21)`
- `#define RGB_WHITE RGB(31, 31, 31)`
- `#define RGB_LIGHTFLESH RGB(30, 20, 15)`
- `#define RGB_BROWN RGB(10, 10, 0)`
- `#define RGB_ORANGE RGB(30, 20, 0)`
- `#define RGB_TEAL RGB(15, 15, 0)`

Typedefs

- typedef `uint16_t palette_color_t`

Functions

- void `set_bkg_palette (uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) OLDCALL`
- void `set_sprite_palette (uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) OLDCALL`
- void `set_bkg_palette_entry (uint8_t palette, uint8_t entry, uint16_t rgb_data) OLDCALL`
- void `set_sprite_palette_entry (uint8_t palette, uint8_t entry, uint16_t rgb_data) OLDCALL`
- void `cpu_slow ()`
- void `cpu_fast ()`
- void `set_default_palette ()`
- void `cgb_compatibility ()`

20.35.1 Detailed Description

Support for the Color GameBoy (CGB).

Enabling CGB features

To unlock and use CGB features and registers you need to change byte 0143h in the cartridge header. Otherwise, the CGB will operate in monochrome "Non CGB" compatibility mode.

- Use a value of **80h** for games that support CGB and monochrome gameboys
(with Lcc: **-Wm-yc**, or makebin directly: **-yc**)
- Use a value of **C0h** for CGB only games.
(with Lcc: **-Wm-yC**, or makebin directly: **-yC**)

See the Pan Docs for more information CGB features.

20.35.2 Macro Definition Documentation

20.35.2.1 RGB `#define RGB(`

```

    r,
    g,
    b ) ((uint16_t) (((b) & 0x1f) << 10) | (((g) & 0x1f) << 5) | (((r) & 0x1f) <<
0)))

```

Macro to create a CGB palette color entry out of 5-bit color components.

Parameters

<i>r</i>	5-bit Red Component, range 0 - 31 (31 brightest)
<i>g</i>	5-bit Green Component, range 0 - 31 (31 brightest)
<i>b</i>	5-bit Blue Component, range 0 - 31 (31 brightest)

The resulting format is bitpacked BGR-555 in a `uint16_t`.

See also

[set_bkg_palette\(\)](#), [set_sprite_palette\(\)](#), [RGB8\(\)](#), [RGBHTML\(\)](#)

20.35.2.2 RGB8 `#define RGB8(`
`r,`
`g,`
`b) ((uint16_t) (((r) >> 3) & 0x1f) | (((g) >> 3) & 0x1f) << 5) | (((b) >>`
`3) & 0x1f) << 10)))`

Macro to create a CGB palette color entry out of 8-bit color components.

Parameters

<i>r</i>	8-bit Red Component, range 0 - 255 (255 brightest)
<i>g</i>	8-bit Green Component, range 0 - 255 (255 brightest)
<i>b</i>	8-bit Blue Component, range 0 - 255 (255 brightest)

The resulting format is bitpacked BGR-555 in a `uint16_t`.

The lowest 3 bits of each color component are dropped during conversion.

See also

[set_bkg_palette\(\)](#), [set_sprite_palette\(\)](#), [RGB\(\)](#), [RGBHTML\(\)](#)

20.35.2.3 RGBHTML `#define RGBHTML(`
`RGB24bit) (RGB8(((RGB24bit) >> 16) & 0xff), ((RGB24bit) >> 8) & 0xff), ((R↔`
`GB24bit) & 0xff)))`

Macro to convert a 24 Bit RGB color to a CGB palette color entry.

Parameters

<i>RGB24bit</i>	Bit packed RGB-888 color (0-255 for each color component).
-----------------	--

The resulting format is bitpacked BGR-555 in a `uint16_t`.

The lowest 3 bits of each color component are dropped during conversion.

See also

[set_bkg_palette\(\)](#), [set_sprite_palette\(\)](#), [RGB\(\)](#), [RGB8\(\)](#)

20.35.2.4 RGB_RED `#define RGB_RED RGB(31, 0, 0)`

Common colors based on the EGA default palette.

20.35.2.5 RGB_DARKRED `#define RGB_DARKRED RGB(15, 0, 0)`

20.35.2.6 RGB_GREEN `#define RGB_GREEN RGB(0, 31, 0)`

20.35.2.7 RGB_DARKGREEN `#define RGB_DARKGREEN RGB(0, 15, 0)`

20.35.2.8 RGB_BLUE `#define RGB_BLUE RGB(0, 0, 31)`

20.35.2.9 RGB_DARKBLUE `#define RGB_DARKBLUE RGB(0, 0, 15)`

20.35.2.10 RGB_YELLOW `#define RGB_YELLOW RGB(31, 31, 0)`

20.35.2.11 RGB_DARKYELLOW `#define RGB_DARKYELLOW RGB(21, 21, 0)`

20.35.2.12 RGB_CYAN `#define RGB_CYAN RGB(0, 31, 31)`

20.35.2.13 RGB_AQUA `#define RGB_AQUA RGB(28, 5, 22)`

20.35.2.14 RGB_PINK `#define RGB_PINK RGB(31, 0, 31)`

20.35.2.15 RGB_PURPLE `#define RGB_PURPLE RGB(21, 0, 21)`

20.35.2.16 RGB_BLACK `#define RGB_BLACK RGB(0, 0, 0)`

20.35.2.17 RGB_DARKGRAY `#define RGB_DARKGRAY RGB(10, 10, 10)`

20.35.2.18 RGB_LIGHTGRAY `#define RGB_LIGHTGRAY RGB(21, 21, 21)`

20.35.2.19 RGB_WHITE `#define RGB_WHITE RGB(31, 31, 31)`

20.35.2.20 RGB_LIGHTFLESH `#define RGB_LIGHTFLESH RGB(30, 20, 15)`

20.35.2.21 RGB_BROWN `#define RGB_BROWN RGB(10, 10, 0)`

20.35.2.22 RGB_ORANGE `#define RGB_ORANGE RGB(30, 20, 0)`

20.35.2.23 RGB_TEAL `#define RGB_TEAL RGB(15, 15, 0)`

20.35.3 Typedef Documentation

20.35.3.1 palette_color_t `typedef uint16_t palette_color_t`
16 bit color entry

20.35.4 Function Documentation

20.35.4.1 set_bkg_palette() `void set_bkg_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

Set CGB background palette(s).

Parameters

<i>first_palette</i>	Index of the first palette to write (0-7)
<i>nb_palettes</i>	Number of palettes to write (1-8, max depends on first_palette)
<i>rgb_data</i>	Pointer to source palette data

Writes **nb_palettes** to background palette data starting at **first_palette**, Palette data is sourced from **rgb_data**.

- Each Palette is 8 bytes in size: 4 colors x 2 bytes per palette color entry.
- Each color (4 per palette) is packed as BGR-555 format (1:5:5:5, MSBit [15] is unused).
- Each component (R, G, B) may have values from 0 - 31 (5 bits), 31 is brightest.

See also

[RGB\(\)](#), [set_bkg_palette_entry\(\)](#)
[BKGF_CGB_PAL0](#), [BKGF_CGB_PAL1](#), [BKGF_CGB_PAL2](#), [BKGF_CGB_PAL3](#)
[BKGF_CGB_PAL4](#), [BKGF_CGB_PAL5](#), [BKGF_CGB_PAL6](#), [BKGF_CGB_PAL7](#)

20.35.4.2 set_sprite_palette() `void set_sprite_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

Set CGB sprite palette(s).

Parameters

<i>first_palette</i>	Index of the first palette to write (0-7)
<i>nb_palettes</i>	Number of palettes to write (1-8, max depends on first_palette)
<i>rgb_data</i>	Pointer to source palette data

Writes **nb_palettes** to sprite palette data starting at **first_palette**, Palette data is sourced from **rgb_data**.

- Each Palette is 8 bytes in size: 4 colors x 2 bytes per palette color entry.
- Each color (4 per palette) is packed as BGR-555 format (1:5:5:5, MSBit [15] is unused).
- Each component (R, G, B) may have values from 0 - 31 (5 bits), 31 is brightest.

See also

[RGB\(\)](#), [set_sprite_palette_entry\(\)](#)
[OAMF_CGB_PAL0](#), [OAMF_CGB_PAL1](#), [OAMF_CGB_PAL2](#), [OAMF_CGB_PAL3](#)
[OAMF_CGB_PAL4](#), [OAMF_CGB_PAL5](#), [OAMF_CGB_PAL6](#), [OAMF_CGB_PAL7](#)

20.35.4.3 set_bkg_palette_entry() `void set_bkg_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `uint16_t rgb_data)`

Sets a single color in the specified CGB background palette.

Parameters

<i>palette</i>	Index of the palette to modify (0-7)
<i>entry</i>	Index of color in palette to modify (0-3)
<i>rgb_data</i>	New color data in BGR 15bpp format.

See also

[set_bkg_palette\(\)](#), [RGB\(\)](#)
[BKGF_CGB_PAL0](#), [BKGF_CGB_PAL1](#), [BKGF_CGB_PAL2](#), [BKGF_CGB_PAL3](#)
[BKGF_CGB_PAL4](#), [BKGF_CGB_PAL5](#), [BKGF_CGB_PAL6](#), [BKGF_CGB_PAL7](#)

20.35.4.4 set_sprite_palette_entry() `void set_sprite_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `uint16_t rgb_data)`

Sets a single color in the specified CGB sprite palette.

Parameters

<i>palette</i>	Index of the palette to modify (0-7)
<i>entry</i>	Index of color in palette to modify (0-3)
<i>rgb_data</i>	New color data in BGR 15bpp format.

See also

[set_sprite_palette\(\)](#), [RGB\(\)](#)
[OAMF_CGB_PAL0](#), [OAMF_CGB_PAL1](#), [OAMF_CGB_PAL2](#), [OAMF_CGB_PAL3](#)
[OAMF_CGB_PAL4](#), [OAMF_CGB_PAL5](#), [OAMF_CGB_PAL6](#), [OAMF_CGB_PAL7](#)

20.35.4.5 cpu_slow() `void cpu_slow ()`

Set CPU speed to slow (Normal Speed) operation.

Interrupts are temporarily disabled and then re-enabled during this call.

In this mode the CGB operates at the same speed as the DMG/Pocket/SGB models.

- You can check to see if `_cpu == CGB_TYPE` before using this function.

See also

[cpu_fast\(\)](#)

20.35.4.6 cpu_fast() `void cpu_fast () [inline]`

Set CPU speed to fast (CGB Double Speed) operation.

On startup the CGB operates in Normal Speed Mode and can be switched into Double speed mode (faster processing but also higher power consumption). See the Pan Docs for more information about which hardware features operate faster and which remain at Normal Speed.

- Interrupts are temporarily disabled and then re-enabled during this call.
- You can check to see if `_cpu == CGB_TYPE` before using this function.

See also

`cpu_slow()`, `_cpu`

20.35.4.7 `set_default_palette()` `void set_default_palette ()`

Set palette, compatible with the DMG/GBP.

The default/first CGB palettes for sprites and backgrounds are set to a similar default appearance as on the DMG/Pocket/SGB models. (White, Light Gray, Dark Gray, Black)

- You can check to see if `_cpu == CGB_TYPE` before using this function.

20.35.4.8 `cgb_compatibility()` `void cgb_compatibility () [inline]`

This function is obsolete

20.36 `gbdk-lib/include/gb/crash_handler.h` File Reference

Functions

- `void __HandleCrash ()`

20.36.1 Detailed Description

When `crash_handler.h` is included, a crash dump screen will be displayed if the CPU executes uninitialized memory (with a value of `0xFF`, the opcode for `RST 38`). A handler is installed for `RST 38` that calls `__HandleCrash()`.

```
#include <gb/crash_handler.h>
```

Also see the `crash` example project included with `gbdk`.

20.36.2 Function Documentation

20.36.2.1 `__HandleCrash()` `void __HandleCrash ()`

Display the crash dump screen.

See the intro for this file for more details.

20.37 `gbdk-lib/include/gb/drawing.h` File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define GRAPHICS_WIDTH 160`
- `#define GRAPHICS_HEIGHT 144`
- `#define SOLID 0x00 /* Overwrites the existing pixels */`
- `#define OR 0x01 /* Performs a logical OR */`
- `#define XOR 0x02 /* Performs a logical XOR */`
- `#define AND 0x03 /* Performs a logical AND */`
- `#define WHITE 0`
- `#define LTGREY 1`

- `#define DKGREY 2`
- `#define BLACK 3`
- `#define M_NOFILL 0`
- `#define M_FILL 1`
- `#define SIGNED 1`
- `#define UNSIGNED 0`

Functions

- void `gprint` (char *str) **NONBANKED**
- void `gprintln` (int16_t number, int8_t radix, int8_t signed_value) **NONBANKED**
- void `gprintn` (int8_t number, int8_t radix, int8_t signed_value) **NONBANKED**
- int8_t `gprintf` (char *fmt,...) **NONBANKED**
- void `plot` (uint8_t x, uint8_t y, uint8_t colour, uint8_t mode) **OLDCALL**
- void `plot_point` (uint8_t x, uint8_t y) **OLDCALL**
- void `switch_data` (uint8_t x, uint8_t y, uint8_t *src, uint8_t *dst) **OLDCALL**
- void `draw_image` (uint8_t *data) **OLDCALL**
- void `line` (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2) **OLDCALL**
- void `box` (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t style) **OLDCALL**
- void `circle` (uint8_t x, uint8_t y, uint8_t radius, uint8_t style) **OLDCALL**
- uint8_t `getpix` (uint8_t x, uint8_t y) **OLDCALL**
- void `wrtchr` (char chr) **OLDCALL**
- void `gotogxy` (uint8_t x, uint8_t y) **OLDCALL**
- void `color` (uint8_t forecolor, uint8_t backcolor, uint8_t mode) **OLDCALL**

20.37.1 Detailed Description

All Points Addressable (APA) mode drawing library.

Drawing routines originally by Pascal Felber Legendary overhaul by Jon Fuge : <https://github.com/jf1452> Commenting by Michael Hope

Note: The standard text `printf()` and `putchar()` cannot be used in APA mode - use `gprintf()` and `wrtchr()` instead.

Note: Using drawing.h will cause it's custom VBL and LCD ISRs (`drawing_vbl` and `drawing_lcd`) to be installed. Changing the mode (`mode(M_TEXT_OUT);`) will cause them to be de-installed.

The valid coordinate ranges are from (x,y) 0,0 to 159,143. There is no built-in clipping, so drawing outside valid coordinates will likely produce undesired results (wrapping/etc).

Important note for the drawing API :

The Game Boy graphics hardware is not well suited to frame-buffer style graphics such as the kind provided in `drawing.h`. Due to that, **most drawing functions (rectangles, circles, etc) will be slow** . When possible it's much faster and more efficient to work with the tiles and tile maps that the Game Boy hardware is built around.

20.37.2 Macro Definition Documentation

20.37.2.1 GRAPHICS_WIDTH `#define GRAPHICS_WIDTH 160`

Size of the screen in pixels

20.37.2.2 GRAPHICS_HEIGHT `#define GRAPHICS_HEIGHT 144`

20.37.2.3 SOLID `#define SOLID 0x00 /* Overwrites the existing pixels */`

20.37.2.4 OR `#define OR 0x01 /* Performs a logical OR */`

20.37.2.5 XOR `#define XOR 0x02 /* Performs a logical XOR */`

20.37.2.6 AND `#define AND 0x03 /* Performs a logical AND */`

20.37.2.7 WHITE `#define WHITE 0`
Possible drawing colours

20.37.2.8 LTGREY `#define LTGREY 1`

20.37.2.9 DKGREY `#define DKGREY 2`

20.37.2.10 BLACK `#define BLACK 3`

20.37.2.11 M_NOFILL `#define M_NOFILL 0`
Possible fill styles for [box\(\)](#) and [circle\(\)](#)

20.37.2.12 M_FILL `#define M_FILL 1`

20.37.2.13 SIGNED `#define SIGNED 1`
Possible values for `signed_value` in [gprintln\(\)](#) and [gprintn\(\)](#)

20.37.2.14 UNSIGNED `#define UNSIGNED 0`

20.37.3 Function Documentation

20.37.3.1 gprint() `void gprint (
char * str)`

Print the string '*str*' with no interpretation

See also

[gotogxy\(\)](#)

20.37.3.2 gprintln() `void gprintln (
int16_t number,
int8_t radix,
int8_t signed_value)`

Print 16 bit **number** in **radix** (base) in the default font at the current text position.

Parameters

<i>number</i>	number to print
<i>radix</i>	radix (base) to print with
<i>signed_value</i>	should be set to SIGNED or UNSIGNED depending on whether the number is signed or not

The current position is advanced by the number of characters printed.

See also

[gotogxy\(\)](#)

20.37.3.3 gprintn() `void gprintn (`
 `int8_t number,`
 `int8_t radix,`
 `int8_t signed_value)`

Print 8 bit **number** in **radix** (base) in the default font at the current text position.

See also

[gprintln\(\)](#), [gotogxy\(\)](#)

20.37.3.4 gprintf() `int8_t gprintf (`
 `char * fmt,`
 `...)`

Print the string and arguments given by **fmt** with arguments ____

Parameters

<i>fmt</i>	The format string as per printf
...	params

Currently supported:

- %c (character)
- %u (int)
- %d (int8_t)
- %o (int8_t as octal)
- %x (int8_t as hex)
- %s (string)

Returns

Returns the number of items printed, or -1 if there was an error.

See also

[gotogxy\(\)](#)

20.37.3.5 plot() `void plot (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t colour,`
 `uint8_t mode)`

Old style plot - try [plot_point\(\)](#)

20.37.3.6 plot_point() `void plot_point (`
 `uint8_t x,`
 `uint8_t y)`

Plot a point in the current drawing mode and colour at **x,y**

20.37.3.7 switch_data() `void switch_data (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t * src,`
 `uint8_t * dst)`

Exchanges the tile on screen at x,y with the tile pointed by src, original tile is saved in dst. Both src and dst may be NULL - saving or copying to screen is not performed in this case.

20.37.3.8 draw_image() `void draw_image (`
 `uint8_t * data)`

Draw a full screen image at **data**

20.37.3.9 line() `void line (`
 `uint8_t x1,`
 `uint8_t y1,`
 `uint8_t x2,`
 `uint8_t y2)`

Draw a line in the current drawing mode and colour from **x1,y1** to **x2,y2**

20.37.3.10 box() `void box (`
 `uint8_t x1,`
 `uint8_t y1,`
 `uint8_t x2,`
 `uint8_t y2,`
 `uint8_t style)`

Draw a box (rectangle) with corners **x1,y1** and **x2,y2** using fill mode **style** (one of NOFILL or FILL)

20.37.3.11 circle() `void circle (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t radius,`
 `uint8_t style)`

Draw a circle with centre at **x,y** and **radius** using fill mode **style** (one of NOFILL or FILL)

20.37.3.12 getpix() `uint8_t getpix (`
 `uint8_t x,`
 `uint8_t y)`

Returns the current colour of the pixel at **x,y**

20.37.3.13 wrtchr() `void wrtchr (`
 `char chr)`

Prints the character **chr** in the default font at the current text position.
The current position is advanced by 1 after the character is printed.

See also

[gotogxy\(\)](#)

20.37.3.14 gotogxy() `void gotogxy (`
 `uint8_t x,`
 `uint8_t y)`

Sets the current text position to **x,y**.

Note: **x** and **y** have units of tiles (8 pixels per unit)

See also

[wrtchr\(\)](#)

20.37.3.15 color() void color (
 uint8_t forecolor,
 uint8_t backcolor,
 uint8_t mode)

Set the current **forecolor** colour, **backcolor** colour, and draw **mode**

Parameters

<i>forecolor</i>	The primary drawing color (outlines of rectangles with box() , letter color with gprintf() , etc).
<i>backcolor</i>	Secondary or background color where applicable (fill color of rectangles with box() when M_FILL is specified, background color of text with gprintf() , etc).
<i>mode</i>	Drawing style to use. Several settings are available SOLID , OR , XOR , AND .

In order to completely overwrite existing pixels use [SOLID](#) for **mode**

20.38 gbdk-lib/include/gb/emu_debug.h File Reference

```
#include <gbdk/emu_debug.h>
```

20.38.1 Detailed Description

Shim for legacy use of [gb/emu_debug.h](#) which has been migrated to [gbdk/emu_debug.h](#)
 See the [emu_debug](#) example project included with gbdk.

20.39 gbdk-lib/include/gbdk/emu_debug.h File Reference

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

Macros

- #define [EMU_MESSAGE](#)(message_text) [EMU_MESSAGE1](#)([EMU_MACRONAME](#)(__LINE__), message_↵
text)
- #define [BGB_MESSAGE](#)(message_text) [EMU_MESSAGE](#)(message_text)
- #define [EMU_PROFILE_BEGIN](#)(MSG) [EMU_MESSAGE_SUFFIX](#)(MSG, "%ZEROCLOCKS%");
- #define [BGB_PROFILE_BEGIN](#)(MSG) [EMU_PROFILE_BEGIN](#)(MSG)
- #define [EMU_TEXT](#)(MSG) [EMU_MESSAGE](#)(MSG)
- #define [BGB_TEXT](#)(MSG) [EMU_TEXT](#)(MSG)
- #define [BGB_printf](#)(...) [EMU_printf](#)(__VA_ARGS__)
- #define [EMU_BREAKPOINT](#) __asm__ ("ld b, b");
- #define [BGB_BREAKPOINT](#) [EMU_BREAKPOINT](#)

Functions

- void [EMU_printf](#) (const char *format,...) [OLDCALL](#)

20.39.1 Detailed Description

Debug window logging and profiling support for emulators (BGB, Emulicious, etc).

Also see the [emu_debug](#) example project included with gbdk.

See the BGB Manual for more information ("expressions, breakpoint conditions, and debug messages") [http↵
://bgb.bircd.org/manual.html#expressions](http://bgb.bircd.org/manual.html#expressions)

20.39.2 Macro Definition Documentation

20.39.2.1 EMU_MESSAGE `#define EMU_MESSAGE(
 message_text) EMU_MESSAGE1(EMU_MACRONAME(__LINE__), message_text)`

Macro to display a message in the emulator debug message window

Parameters

<code>message_text</code>	Quoted text string to display in the debug message window
---------------------------	---

The following special parameters can be used when bracketed with "%" characters.

- CPU registers: AF, BC, DE, HL, SP, PC, B, C, D, E, H, L, A, ZERO, ZF, Z, CARRY, CY, IME, ALLREGS
- Other state values: ROMBANK, XRAMBANK, SRAMBANK, WRAMBANK, VRAMBANK, TOTALCLKS, LA↵STCLKS, CLK2VBLANK

Example: print a message along with the currently active ROM bank.

```
EMU_MESSAGE("Current ROM Bank is: %ROMBANK%");
```

See the BGB Manual for more information ("expressions, breakpoint conditions, and debug messages") [http↵://bgb.bircd.org/manual.html#expressions](http://bgb.bircd.org/manual.html#expressions)

See also

[EMU_PROFILE_BEGIN\(\)](#), [EMU_PROFILE_END\(\)](#)

20.39.2.2 BGB_MESSAGE `#define BGB_MESSAGE(
 message_text) EMU_MESSAGE(message_text)`

20.39.2.3 EMU_PROFILE_BEGIN `#define EMU_PROFILE_BEGIN(
 MSG) EMU_MESSAGE_SUFFIX(MSG, "%ZEROCCLKS%");`

Macro to **Start** a profiling block for the emulator (BGB, Emulicious, etc)

Parameters

<code>MSG</code>	Quoted text string to display in the debug message window along with the result
------------------	---

To complete the profiling block and print the result call `EMU_PROFILE_END`.

See also

`EMU_PROFILE_END()`, [EMU_MESSAGE\(\)](#)

20.39.2.4 BGB_PROFILE_BEGIN `#define BGB_PROFILE_BEGIN(
 MSG) EMU_PROFILE_BEGIN(MSG)`

20.39.2.5 EMU_TEXT `#define EMU_TEXT(
 MSG) EMU_MESSAGE(MSG)`

Macro to **End** a profiling block and print the results in the emulator debug message window

Parameters

<code>MSG</code>	Quoted text string to display in the debug message window along with the result
------------------	---

This should only be called after a previous call to [EMU_PROFILE_BEGIN\(\)](#)

The results are in Emulator clock units, which are "1 nop in [CGB] doublespeed mode".

So when running in Normal Speed mode (i.e. non-CGB doublespeed) the printed result should be **divided by 2** to get the actual elapsed cycle count.

If running in CB Double Speed mode use the below call instead, it correctly compensates for the speed difference. In this scenario, the result does **not need to be divided by 2** to get the elapsed cycle count.

```
EMU_MESSAGE("NOP TIME: %~4+LASTCLKS%");
```

See also

[EMU_PROFILE_BEGIN\(\)](#), [EMU_MESSAGE\(\)](#)

20.39.2.6 BGB_TEXT `#define BGB_TEXT(
MSG) EMU_TEXT(MSG)`

20.39.2.7 BGB_printf `#define BGB_printf(
...) EMU_printf(__VA_ARGS__)`

20.39.2.8 EMU_BREAKPOINT `#define EMU_BREAKPOINT __asm__("ld b, b");`
The Emulator will break into debugger when encounters this line

20.39.2.9 BGB_BREAKPOINT `#define BGB_BREAKPOINT EMU_BREAKPOINT`

20.39.3 Function Documentation

20.39.3.1 EMU_printf() `void EMU_printf (
const char * format,
...)`

Display preset debug information in the Emulator debug messages window.

This function is equivalent to:

```
EMU_MESSAGE("PROFILE,%(SP+$0)%,%(SP+$1)%,%A%,%TOTALCLKS%,%ROMBANK%,%WRAMBANK%");
```

Print the string and arguments given by format to the emulator debug message window

Parameters

<i>format</i>	The format string as per printf
---------------	---------------------------------

Does not return the number of characters printed. Result string MUST BE LESS OR EQUAL THAN 128 BYTES LONG, INCLUDING THE TRAILIG ZERO BYTE!

Currently supported:

- %hx (char as hex)
- %hu (unsigned char)
- %hd (signed char)
- %c (character)
- %u (unsigned int)
- %d (signed int)
- %x (unsigned int as hex)
- %s (string)

Warning: to correctly pass chars for printing as chars, they *must* be explicitly re-cast as such when calling the function. See [docs_chars_varargs](#) for more details.

20.40 gbdk-lib/include/gb/gb.h File Reference

```
#include <types.h>
#include <stdint.h>
#include <gbdk/version.h>
#include <gb/hardware.h>
```

Data Structures

- struct [joypads_t](#)
- struct [OAM_item_t](#)

Macros

- #define [NINTENDO](#)
- #define [GAMEBOY](#)
- #define [J_UP](#) 0x04U
- #define [J_DOWN](#) 0x08U
- #define [J_LEFT](#) 0x02U
- #define [J_RIGHT](#) 0x01U
- #define [J_A](#) 0x10U
- #define [J_B](#) 0x20U
- #define [J_SELECT](#) 0x40U
- #define [J_START](#) 0x80U
- #define [M_DRAWING](#) 0x01U
- #define [M_TEXT_OUT](#) 0x02U
- #define [M_TEXT_INOUT](#) 0x03U
- #define [M_NO_SCROLL](#) 0x04U
- #define [M_NO_INTERP](#) 0x08U
- #define [S_PALETTE](#) 0x10U
- #define [S_FLIPX](#) 0x20U
- #define [S_FLIPY](#) 0x40U
- #define [S_PRIORITY](#) 0x80U
- #define [EMPTY_IFLAG](#) 0x00U
- #define [VBL_IFLAG](#) 0x01U
- #define [LCD_IFLAG](#) 0x02U
- #define [TIM_IFLAG](#) 0x04U
- #define [SIO_IFLAG](#) 0x08U
- #define [JOY_IFLAG](#) 0x10U
- #define [DMG_BLACK](#) 0x03
- #define [DMG_DARK_GRAY](#) 0x02
- #define [DMG_LITE_GRAY](#) 0x01
- #define [DMG_WHITE](#) 0x00
- #define [DMG_PALETTE](#)(C0, C1, C2, C3) (((uint8_t) (((C3) & 0x03) << 6) | (((C2) & 0x03) << 4) | (((C1) & 0x03) << 2) | ((C0) & 0x03)))
- #define [SCREENWIDTH](#) [DEVICE_SCREEN_PX_WIDTH](#)
- #define [SCREENHEIGHT](#) [DEVICE_SCREEN_PX_HEIGHT](#)
- #define [MINWNDPOSX](#) 0x07U
- #define [MINWNDPOSY](#) 0x00U
- #define [MAXWNDPOSX](#) 0xA6U
- #define [MAXWNDPOSY](#) 0x8FU
- #define [DMG_TYPE](#) 0x01
- #define [MGB_TYPE](#) 0xFF
- #define [CGB_TYPE](#) 0x11
- #define [GBA_NOT_DETECTED](#) 0x00

- `#define GBA_DETECTED 0x01`
- `#define DEVICE_SUPPORTS_COLOR (_cpu == CGB_TYPE)`
- `#define IO_IDLE 0x00U`
- `#define IO_SENDING 0x01U`
- `#define IO_RECEIVING 0x02U`
- `#define IO_ERROR 0x04U`
- `#define CURRENT_BANK _current_bank`
- `#define BANK(VARNAME) ((uint8_t) & __bank_ ## VARNAME)`
- `#define BANKREF(VARNAME)`
- `#define BANKREF_EXTERN(VARNAME) extern const void __bank_ ## VARNAME;`
- `#define SWITCH_ROM_MEGADUCK(b) _current_bank = (b), *(volatile uint8_t *)0x0001 = (b)`
- `#define SWITCH_ROM_MBC1(b) _current_bank = (b), *(volatile uint8_t *)0x2000 = (b)`
- `#define SWITCH_ROM SWITCH_ROM_MBC1`
- `#define SWITCH_RAM_MBC1(b) *(volatile uint8_t *)0x4000 = (b)`
- `#define SWITCH_RAM SWITCH_RAM_MBC1`
- `#define ENABLE_RAM_MBC1 *(volatile uint8_t *)0x0000 = 0x0A`
- `#define ENABLE_RAM ENABLE_RAM_MBC1`
- `#define DISABLE_RAM_MBC1 *(volatile uint8_t *)0x0000 = 0x00`
- `#define DISABLE_RAM DISABLE_RAM_MBC1`
- `#define SWITCH_16_8_MODE_MBC1 *(volatile uint8_t *)0x6000 = 0x00`
- `#define SWITCH_4_32_MODE_MBC1 *(volatile uint8_t *)0x6000 = 0x01`
- `#define SWITCH_ROM_MBC5(b)`
- `#define SWITCH_ROM_MBC5_8M(b)`
- `#define SWITCH_RAM_MBC5(b) *(volatile uint8_t *)0x4000 = (b)`
- `#define ENABLE_RAM_MBC5 *(volatile uint8_t *)0x0000 = 0x0A`
- `#define DISABLE_RAM_MBC5 *(volatile uint8_t *)0x0000 = 0x00`
- `#define DISPLAY_ON LCDC_REG|=LCDCF_ON`
- `#define DISPLAY_OFF display_off();`
- `#define HIDE_LEFT_COLUMN`
- `#define SHOW_LEFT_COLUMN`
- `#define SHOW_BKG LCDC_REG|=LCDCF_BGON`
- `#define HIDE_BKG LCDC_REG&=~LCDCF_BGON`
- `#define SHOW_WIN LCDC_REG|=LCDCF_WINON`
- `#define HIDE_WIN LCDC_REG&=~LCDCF_WINON`
- `#define SHOW_SPRITES LCDC_REG|=LCDCF_OBJON`
- `#define HIDE_SPRITES LCDC_REG&=~LCDCF_OBJON`
- `#define SPRITES_8x16 LCDC_REG|=LCDCF_OBJ16`
- `#define SPRITES_8x8 LCDC_REG&=~LCDCF_OBJ16`
- `#define COMPAT_PALETTE(C0, C1, C2, C3) (((uint8_t)((C3) << 6) | ((C2) << 4) | ((C1) << 2) | (C0)))`
- `#define set_bkg_2bpp_data set_bkg_data`
- `#define set_tile_map set_bkg_tiles`
- `#define set_tile_submap set_bkg_submap`
- `#define set_tile_xy set_bkg_tile_xy`
- `#define set_sprite_2bpp_data set_sprite_data`
- `#define DISABLE_OAM_DMA_shadow_OAM_base = 0`
- `#define DISABLE_VBL_TRANSFER DISABLE_OAM_DMA`
- `#define ENABLE_OAM_DMA_shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM >> 8)`
- `#define ENABLE_VBL_TRANSFER ENABLE_OAM_DMA`
- `#define MAX_HARDWARE_SPRITES 40`
- `#define fill_rect fill_bkg_rect`

Typedefs

- `typedef void(* int_handler) (void) NONBANKED`
- `typedef struct OAM_item_t OAM_item_t`

Functions

- void `remove_VBL` (int_handler h) OLDCALL
- void `remove_LCD` (int_handler h) OLDCALL
- void `remove_TIM` (int_handler h) OLDCALL
- void `remove_SIO` (int_handler h) OLDCALL
- void `remove_JOY` (int_handler h) OLDCALL
- void `add_VBL` (int_handler h) OLDCALL
- void `add_LCD` (int_handler h) OLDCALL
- void `add_TIM` (int_handler h) OLDCALL
- void `add_low_priority_TIM` (int_handler h) OLDCALL
- void `add_SIO` (int_handler h) OLDCALL
- void `add_JOY` (int_handler h) OLDCALL
- void `nowait_int_handler` ()
- void `wait_int_handler` ()
- uint8_t `cancel_pending_interrupts` ()
- void `mode` (uint8_t m) OLDCALL
- uint8_t `get_mode` () OLDCALL PRESERVES_REGS(b)
- void `send_byte` ()
- void `receive_byte` ()
- void `delay` (uint16_t d) PRESERVES_REGS(h)
- uint8_t `joypad` () PRESERVES_REGS(b)
- uint8_t `waitpad` (uint8_t mask) PRESERVES_REGS(b)
- void `waitpadup` () PRESERVES_REGS(a)
- uint8_t `joypad_init` (uint8_t npads, joypads_t *joypads) OLDCALL
- void `joypad_ex` (joypads_t *joypads) PRESERVES_REGS(b)
- void `enable_interrupts` () PRESERVES_REGS(a)
- void `disable_interrupts` () PRESERVES_REGS(a)
- void `set_interrupts` (uint8_t flags) OLDCALL PRESERVES_REGS(b)
- void `reset` ()
- void `wait_vbl_done` () PRESERVES_REGS(b)
- void `display_off` () PRESERVES_REGS(b)
- void `refresh_OAM` () PRESERVES_REGS(b)
- void `hramcpy` (uint8_t dst, const void *src, uint8_t n) OLDCALL PRESERVES_REGS(b)
- void `set_vram_byte` (uint8_t *addr, uint8_t v) OLDCALL PRESERVES_REGS(b)
- uint8_t `get_vram_byte` (uint8_t *addr) PRESERVES_REGS(b)
- uint8_t * `get_bkg_xy_addr` (uint8_t x, uint8_t y) OLDCALL PRESERVES_REGS(b)
- void `set_2bpp_palette` (uint16_t palette)
- void `set_1bpp_colors_ex` (uint8_t fgcolor, uint8_t bgcolor, uint8_t mode) OLDCALL
- void `set_1bpp_colors` (uint8_t fgcolor, uint8_t bgcolor)
- void `set_bkg_data` (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) OLDCALL PRESERVES_REGS(b)
- void `set_bkg_1bpp_data` (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) OLDCALL PRESERVES_REGS(b)
- void `get_bkg_data` (uint8_t first_tile, uint8_t nb_tiles, uint8_t *data) OLDCALL PRESERVES_REGS(b)
- void `set_bkg_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles) OLDCALL PRESERVES_REGS(b)
- void `set_bkg_based_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void `set_bkg_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w) OLDCALL
- void `set_bkg_based_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void `get_bkg_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t *tiles) OLDCALL PRESERVES_REGS(b)
- uint8_t * `set_bkg_tile_xy` (uint8_t x, uint8_t y, uint8_t t) OLDCALL PRESERVES_REGS(b)
- uint8_t `get_bkg_tile_xy` (uint8_t x, uint8_t y) OLDCALL PRESERVES_REGS(b)
- void `move_bkg` (uint8_t x, uint8_t y)
- void `scroll_bkg` (int8_t x, int8_t y)
- uint8_t * `get_win_xy_addr` (uint8_t x, uint8_t y) OLDCALL PRESERVES_REGS(b)
- void `set_win_data` (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) OLDCALL PRESERVES_REGS(b)

- void [set_win_1bpp_data](#) (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) [OLDCALL PRESERVES_REGS\(b](#)
- void [get_win_data](#) (uint8_t first_tile, uint8_t nb_tiles, uint8_t *data) [OLDCALL PRESERVES_REGS\(b](#)
- void [set_win_tiles](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles) [OLDCALL PRESERVES_REGS\(b](#)
- void [set_win_based_tiles](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void [set_win_submap](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w) [OLDCALL](#)
- void [set_win_based_submap](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void [get_win_tiles](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t *tiles) [OLDCALL PRESERVES_REGS\(b](#)
- uint8_t * [set_win_tile_xy](#) (uint8_t x, uint8_t y, uint8_t t) [OLDCALL PRESERVES_REGS\(b](#)
- uint8_t [get_win_tile_xy](#) (uint8_t x, uint8_t y) [OLDCALL PRESERVES_REGS\(b](#)
- void [move_win](#) (uint8_t x, uint8_t y)
- void [scroll_win](#) (uint8_t x, uint8_t y)
- void [set_sprite_data](#) (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) [OLDCALL PRESERVES_REGS\(b](#)
- void [set_sprite_1bpp_data](#) (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data) [OLDCALL PRESERVES_REGS\(b](#)
- void [get_sprite_data](#) (uint8_t first_tile, uint8_t nb_tiles, uint8_t *data) [OLDCALL PRESERVES_REGS\(b](#)
- void [SET_SHADOW_OAM_ADDRESS](#) (void *address)
- void [set_sprite_tile](#) (uint8_t nb, uint8_t tile)
- uint8_t [get_sprite_tile](#) (uint8_t nb)
- void [set_sprite_prop](#) (uint8_t nb, uint8_t prop)
- uint8_t [get_sprite_prop](#) (uint8_t nb)
- void [move_sprite](#) (uint8_t nb, uint8_t x, uint8_t y)
- void [scroll_sprite](#) (uint8_t nb, uint8_t x, uint8_t y)
- void [hide_sprite](#) (uint8_t nb)
- void [set_data](#) (uint8_t *vram_addr, const uint8_t *data, uint16_t len) [OLDCALL PRESERVES_REGS\(b](#)
- void [get_data](#) (uint8_t *data, uint8_t *vram_addr, uint16_t len) [OLDCALL PRESERVES_REGS\(b](#)
- void [memcpy](#) (uint8_t *dest, uint8_t *sour, uint16_t len) [OLDCALL PRESERVES_REGS\(b](#)
- void [set_tiles](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t *vram_addr, const uint8_t *tiles) [OLDCALL](#)
- void [set_tile_data](#) (uint8_t first_tile, uint8_t nb_tiles, const uint8_t *data, uint8_t base) [OLDCALL PRESERVES_REGS\(b](#)
- void [get_tiles](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t *vram_addr, uint8_t *tiles) [OLDCALL](#)
- void [set_native_tile_data](#) (uint16_t first_tile, uint8_t nb_tiles, const uint8_t *data)
- void [init_win](#) (uint8_t c) [OLDCALL PRESERVES_REGS\(b](#)
- void [init_bkg](#) (uint8_t c) [OLDCALL PRESERVES_REGS\(b](#)
- void [memset](#) (void *s, uint8_t c, size_t n) [OLDCALL PRESERVES_REGS\(b](#)
- void [fill_bkg_rect](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t tile) [OLDCALL PRESERVES_REGS\(b](#)
- void [fill_win_rect](#) (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t tile) [OLDCALL PRESERVES_REGS\(b](#)

Variables

- [uint8_t c](#)
- [uint8_t cpu](#)
- [uint8_t is_GBA](#)
- volatile [uint16_t sys_time](#)
- volatile [uint8_t io_status](#)
- volatile [uint8_t io_in](#)
- volatile [uint8_t io_out](#)
- [__REG_current_bank](#)
- void [l](#)
- [uint8_t h](#)
- void [b](#)
- void [d](#)
- void [e](#)
- [uint16_t current_1bpp_colors](#)
- [uint8_t map_tile_offset](#)
- [uint8_t submap_tile_offset](#)
- volatile struct [OAM_item_t shadow_OAM](#) []
- [__REG_shadow_OAM_base](#)

20.40.1 Detailed Description

Gameboy specific functions.

20.40.2 Macro Definition Documentation

20.40.2.1 NINTENDO `#define NINTENDO`

20.40.2.2 GAMEBOY `#define GAMEBOY`

20.40.2.3 J_UP `#define J_UP 0x04U`

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_t keys; keys = joypad(); if (keys & J_B) { ... }
```

See also

[joypad](#)

20.40.2.4 J_DOWN `#define J_DOWN 0x08U`

20.40.2.5 J_LEFT `#define J_LEFT 0x02U`

20.40.2.6 J_RIGHT `#define J_RIGHT 0x01U`

20.40.2.7 J_A `#define J_A 0x10U`

20.40.2.8 J_B `#define J_B 0x20U`

20.40.2.9 J_SELECT `#define J_SELECT 0x40U`

20.40.2.10 J_START `#define J_START 0x80U`

20.40.2.11 M_DRAWING `#define M_DRAWING 0x01U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

20.40.2.12 M_TEXT_OUT `#define M_TEXT_OUT 0x02U`

20.40.2.13 M_TEXT_INOUT `#define M_TEXT_INOUT 0x03U`

20.40.2.14 M_NO_SCROLL `#define M_NO_SCROLL 0x04U`

Set this in addition to the others to disable scrolling

If scrolling is disabled, the cursor returns to (0,0)

See also

[mode\(\)](#)

20.40.2.15 M_NO_INTERP `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

20.40.2.16 S_PALETTE `#define S_PALETTE 0x10U`

If this is set, sprite colours come from OBJ1PAL. Else they come from OBJ0PAL

See also

[set_sprite_prop\(\)](#).

20.40.2.17 S_FLIPX `#define S_FLIPX 0x20U`

If set the sprite will be flipped horizontally.

See also

[set_sprite_prop\(\)](#)

20.40.2.18 S_FLIPY `#define S_FLIPY 0x40U`

If set the sprite will be flipped vertically.

See also

[set_sprite_prop\(\)](#)

20.40.2.19 S_PRIORITY `#define S_PRIORITY 0x80U`

If this bit is clear, then the sprite will be displayed on top of the background and window.

See also

[set_sprite_prop\(\)](#)

20.40.2.20 EMPTY_IFLAG `#define EMPTY_IFLAG 0x00U`

Disable calling of interrupt service routines

20.40.2.21 VBL_IFLAG `#define VBL_IFLAG 0x01U`

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

See also

[set_interrupts\(\)](#),
[add_VBL](#)

20.40.2.22 LCD_IFLAG `#define LCD_IFLAG 0x02U`

LCD Interrupt when triggered by the STAT register.

See also

[set_interrupts\(\)](#),
[add_LCD](#)

20.40.2.23 TIM_IFLAG `#define TIM_IFLAG 0x04U`

Timer Interrupt when the timer [TIMA_REG](#) overflows.

See also

[set_interrupts\(\)](#),
[add_TIM](#)

20.40.2.24 SIO_IFLAG `#define SIO_IFLAG 0x08U`

Serial Link Interrupt occurs when the serial transfer has completed.

See also

[set_interrupts\(\)](#),
[add_SIO](#)

20.40.2.25 JOY_IFLAG `#define JOY_IFLAG 0x10U`

Joypad Interrupt occurs on a transition of the keypad.

See also

[set_interrupts\(\)](#),
[add_JOY](#)

20.40.2.26 DMG_BLACK `#define DMG_BLACK 0x03`**20.40.2.27 DMG_DARK_GRAY** `#define DMG_DARK_GRAY 0x02`**20.40.2.28 DMG_LITE_GRAY** `#define DMG_LITE_GRAY 0x01`

20.40.2.29 DMG_WHITE `#define DMG_WHITE 0x00`

20.40.2.30 DMG_PALETTE `#define DMG_PALETTE(
 C0,
 C1,
 C2,
 C3) ((uint8_t) (((C3) & 0x03) << 6) | (((C2) & 0x03) << 4) | (((C1) & 0x03) <<
 2) | ((C0) & 0x03)))`

Macro to create a DMG palette from 4 colors

Parameters

<i>C0</i>	Color for Index 0
<i>C1</i>	Color for Index 1
<i>C2</i>	Color for Index 2
<i>C3</i>	Color for Index 3

The resulting format is four greyscale colors packed into a single unsigned byte.

Example:

```
BGP_REG = DMG_PALETTE(DMG_BLACK, DMG_DARK_GRAY, DMG_LITE_GRAY, DMG_WHITE);
```

See also

[OBP0_REG](#), [OBP1_REG](#), [BGP_REG](#)

[DMG_BLACK](#), [DMG_DARK_GRAY](#), [DMG_LITE_GRAY](#), [DMG_WHITE](#)

20.40.2.31 SCREENWIDTH `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`

Width of the visible screen in pixels.

20.40.2.32 SCREENHEIGHT `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`

Height of the visible screen in pixels.

20.40.2.33 MINWNDPOSX `#define MINWNDPOSX 0x07U`

The Minimum X position of the Window Layer (Left edge of screen)

See also

[move_win\(\)](#)

20.40.2.34 MINWNDPOSY `#define MINWNDPOSY 0x00U`

The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move_win\(\)](#)

20.40.2.35 MAXWNDPOSX `#define MAXWNDPOSX 0xA6U`

The Maximum X position of the Window Layer (Right edge of screen)

See also

[move_win\(\)](#)

20.40.2.36 MAXWNDPOSY `#define MAXWNDPOSY 0x8FU`

The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move_win\(\)](#)

20.40.2.37 DMG_TYPE `#define DMG_TYPE 0x01`

Hardware Model: Original GB or Super GB.

See also

[_cpu](#)

20.40.2.38 MGB_TYPE `#define MGB_TYPE 0xFF`

Hardware Model: Pocket GB or Super GB 2.

See also

[_cpu](#)

20.40.2.39 CGB_TYPE `#define CGB_TYPE 0x11`

Hardware Model: Color GB.

See also

[_cpu](#)

20.40.2.40 GBA_NOT_DETECTED `#define GBA_NOT_DETECTED 0x00`

Hardware Model: DMG, CGB or MGB.

See also

[_cpu](#), [_is_GBA](#)

20.40.2.41 GBA_DETECTED `#define GBA_DETECTED 0x01`

Hardware Model: GBA.

See also

[_cpu](#), [_is_GBA](#)

20.40.2.42 DEVICE_SUPPORTS_COLOR `#define DEVICE_SUPPORTS_COLOR (_cpu == CGB_TYPE)`

Macro returns TRUE if device supports color

20.40.2.43 IO_IDLE `#define IO_IDLE 0x00U`

Serial Link IO is completed

20.40.2.44 IO_SENDING `#define IO_SENDING 0x01U`

Serial Link Sending data

20.40.2.45 IO_RECEIVING `#define IO_RECEIVING 0x02U`
Serial Link Receiving data

20.40.2.46 IO_ERROR `#define IO_ERROR 0x04U`
Serial Link Error

20.40.2.47 CURRENT_BANK `#define CURRENT_BANK __current_bank`

20.40.2.48 BANK `#define BANK(`
`VARNAME) ((uint8_t) & __bank_ ## VARNAME)`
Obtains the **bank number** of VARNAME

Parameters

<i>VARNAME</i>	Name of the variable which has a <code>__bank_</code> VARNAME companion symbol which is adjusted by bankpack
----------------	--

Use this to obtain the bank number from a bank reference created with [BANKREF\(\)](#).

See also

[BANKREF_EXTERN\(\)](#), [BANKREF\(\)](#)

20.40.2.49 BANKREF `#define BANKREF(`
`VARNAME)`

Value:

```
void __func_ ## VARNAME() __banked __naked { \
__asm \
    .local b__func_ ## VARNAME \
    __bank_ ## VARNAME = b__func_ ## VARNAME \
    .globl __bank_ ## VARNAME \
__endasm; \
}
```

Creates a reference for retrieving the bank number of a variable or function

Parameters

<i>VARNAME</i>	Variable name to use, which may be an existing identifier
----------------	---

See also

[BANK\(\)](#) for obtaining the bank number of the included data.

More than one [BANKREF \(\)](#) may be created per file, but each call should always use a unique VARNAME. Use [BANKREF_EXTERN\(\)](#) within another source file to make the variable and it's data accesible there.

20.40.2.50 BANKREF_EXTERN `#define BANKREF_EXTERN(`
`VARNAME) extern const void __bank_ ## VARNAME;`

Creates extern references for accessing a [BANKREF\(\)](#) generated variable.

Parameters

<i>VARNAME</i>	Name of the variable used with BANKREF()
----------------	--

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

20.40.2.51 SWITCH_ROM_MEGADUCK `#define SWITCH_ROM_MEGADUCK(
 b) _current_bank = (b), *(volatile uint8_t *)0x0001 = (b)`

Makes MEGADUCK MBC switch the active ROM bank

Parameters

<i>b</i>	ROM bank to switch to (max 3 for 64K, or 7 for 128K)
----------	--

20.40.2.52 SWITCH_ROM_MBC1 `#define SWITCH_ROM_MBC1(
 b) _current_bank = (b), *(volatile uint8_t *)0x2000 = (b)`

Makes MBC1 and other compatible MBCs switch the active ROM bank

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

For MBC1 some banks in it's range are unavailable (typically 0x20, 0x40, 0x60).

See pandocs for more details <https://gbdev.io/pandocs/MBC1>

20.40.2.53 SWITCH_ROM `#define SWITCH_ROM SWITCH_ROM_MBC1`

Makes default platform MBC switch the active ROM bank

Parameters

<i>b</i>	ROM bank to switch to (max 255)
----------	---------------------------------

- When used with MBC1 the max bank is Bank 31 (512K).
- When used with MBC5 the max bank is Bank 255 (4MB).
- To use the full 8MB size of MBC5 see [SWITCH_ROM_MBC5_8M\(\)](#).
- For MBC1 some banks in it's range are unavailable (typically 0x20, 0x40, 0x60).

Note: Using [SWITCH_ROM_MBC5_8M\(\)](#) should not be mixed with using [SWITCH_ROM_MBC5\(\)](#) and [SWITCH_ROM\(\)](#).

See also

[SWITCH_ROM_MBC1](#), [SWITCH_ROM_MBC5](#), [SWITCH_ROM_MEGADUCK](#)

20.40.2.54 SWITCH_RAM_MBC1 `#define SWITCH_RAM_MBC1(
 b) *(volatile uint8_t *)0x4000 = (b)`

Switches SRAM bank on MBC1 and other compaticle MBCs

Parameters

<i>b</i>	SRAM bank to switch to
----------	------------------------

20.40.2.55 SWITCH_RAM `#define SWITCH_RAM SWITCH_RAM_MBC1`

Switches SRAM bank on MBC1 and other compaticle MBCs

Parameters

<i>b</i>	SRAM bank to switch to
----------	------------------------

See also

[SWITCH_RAM_MBC1](#), [SWITCH_RAM_MBC5](#)**20.40.2.56 ENABLE_RAM_MBC1** `#define ENABLE_RAM_MBC1 *(volatile uint8_t *)0x0000 = 0x0A`

Enables SRAM on MBC1

20.40.2.57 ENABLE_RAM `#define ENABLE_RAM ENABLE_RAM_MBC1`**20.40.2.58 DISABLE_RAM_MBC1** `#define DISABLE_RAM_MBC1 *(volatile uint8_t *)0x0000 = 0x00`

Disables SRAM on MBC1

20.40.2.59 DISABLE_RAM `#define DISABLE_RAM DISABLE_RAM_MBC1`**20.40.2.60 SWITCH_16_8_MODE_MBC1** `#define SWITCH_16_8_MODE_MBC1 *(volatile uint8_t *)0x6000 = 0x00`**20.40.2.61 SWITCH_4_32_MODE_MBC1** `#define SWITCH_4_32_MODE_MBC1 *(volatile uint8_t *)0x6000 = 0x01`**20.40.2.62 SWITCH_ROM_MBC5** `#define SWITCH_ROM_MBC5(
 b)`

Value:

```

    _current_bank = (b), \
    *(volatile uint8_t *)0x3000 = 0, \
    *(volatile uint8_t *)0x2000 = (b)

```

Makes MBC5 switch to the active ROM bank

Parameters

<i>b</i>	ROM bank to switch to (max 255)
----------	---------------------------------

Supports up to ROM bank 255 (4 MB).

[SWITCH_ROM_MBC5_8M](#) may be used if the full 8MB size is needed.Note: Using [SWITCH_ROM_MBC5_8M\(\)](#) should not be mixed with using [SWITCH_ROM_MBC5\(\)](#) and [SWITCH_ROM\(\)](#).

Note the order used here. Writing the other way around on a MBC1 always selects bank 1

20.40.2.63 SWITCH_ROM_MBC5_8M `#define SWITCH_ROM_MBC5_8M(
 b)`

Value:

```

    *(volatile uint8_t *)0x3000 = ((uint16_t)(b) >> 8), \

```

```
*(volatile uint8_t *)0x2000 = (b)
```

Makes MBC5 to switch the active ROM bank using the full 8MB size.

See also

[_current_bank](#)

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

This is an alternate to [SWITCH_ROM_MBC5](#) which is limited to 4MB.

Note:

- Banked SDCC calls are not supported if you use this macro.
- The active bank number is not tracked by [_current_bank](#) if you use this macro.
- Using [SWITCH_ROM_MBC5_8M\(\)](#) should not be mixed with using [SWITCH_ROM_MBC5\(\)](#) and [SWITCH_ROM\(\)](#).

Note the order used here. Writing the other way around on a MBC1 always selects bank 1

20.40.2.64 SWITCH_RAM_MBC5 `#define SWITCH_RAM_MBC5(
 b) *(volatile uint8_t *)0x4000 = (b)`

Switches SRAM bank on MBC5

Parameters

<i>b</i>	SRAM bank to switch to
----------	------------------------

20.40.2.65 ENABLE_RAM_MBC5 `#define ENABLE_RAM_MBC5 *(volatile uint8_t *)0x0000 = 0x0A`
Enables SRAM on MBC5

20.40.2.66 DISABLE_RAM_MBC5 `#define DISABLE_RAM_MBC5 *(volatile uint8_t *)0x0000 = 0x00`
Disables SRAM on MBC5

20.40.2.67 DISPLAY_ON `#define DISPLAY_ON LCDC_REG|=LCDCF_ON`
Turns the display back on.

See also

[display_off](#), [DISPLAY_OFF](#)

20.40.2.68 DISPLAY_OFF `#define DISPLAY_OFF display_off();`
Turns the display off immediately.

See also

[display_off](#), [DISPLAY_ON](#)

20.40.2.69 HIDE_LEFT_COLUMN `#define HIDE_LEFT_COLUMN`
Does nothing for GB

20.40.2.70 SHOW_LEFT_COLUMN `#define SHOW_LEFT_COLUMN`
Does nothing for GB

20.40.2.71 SHOW_BKG `#define SHOW_BKG LCDC_REG|=LCDCF_BGON`
Turns on the background layer. Sets bit 0 of the LCDC register to 1.

20.40.2.72 HIDE_BKG `#define HIDE_BKG LCDC_REG&=~LCDCF_BGON`
Turns off the background layer. Sets bit 0 of the LCDC register to 0.

20.40.2.73 SHOW_WIN `#define SHOW_WIN LCDC_REG|=LCDCF_WINON`
Turns on the Window layer Sets bit 5 of the LCDC register to 1.
This only controls Window visibility. If either the Background layer (which the window is part of) or the Display are not turned then the Window contents will not be visible. Those can be turned on using [SHOW_BKG](#) and [DISPLAY_ON](#).

20.40.2.74 HIDE_WIN `#define HIDE_WIN LCDC_REG&=~LCDCF_WINON`
Turns off the window layer. Clears bit 5 of the LCDC register to 0.

20.40.2.75 SHOW_SPRITES `#define SHOW_SPRITES LCDC_REG|=LCDCF_OBJON`
Turns on the sprites layer. Sets bit 1 of the LCDC register to 1.

20.40.2.76 HIDE_SPRITES `#define HIDE_SPRITES LCDC_REG&=~LCDCF_OBJON`
Turns off the sprites layer. Clears bit 1 of the LCDC register to 0.

See also

[hide_sprite](#), [hide_sprites_range](#)

20.40.2.77 SPRITES_8x16 `#define SPRITES_8x16 LCDC_REG|=LCDCF_OBJ16`
Sets sprite size to 8x16 pixels, two tiles one above the other. Sets bit 2 of the LCDC register to 1.

20.40.2.78 SPRITES_8x8 `#define SPRITES_8x8 LCDC_REG&=~LCDCF_OBJ16`
Sets sprite size to 8x8 pixels, one tile. Clears bit 2 of the LCDC register to 0.

20.40.2.79 COMPAT_PALETTE `#define COMPAT_PALETTE(
 C0,
 C1,
 C2,
 C3) ((uint8_t)((C3) << 6) | ((C2) << 4) | ((C1) << 2) | (C0))`

20.40.2.80 set_bkg_2bpp_data `#define set_bkg_2bpp_data set_bkg_data`

20.40.2.81 set_tile_map `#define set_tile_map set_bkg_tiles`

20.40.2.82 set_tile_submap `#define set_tile_submap set_bkg_submap`

20.40.2.83 set_tile_xy `#define set_tile_xy set_bkg_tile_xy`

20.40.2.84 set_sprite_2bpp_data `#define set_sprite_2bpp_data set_sprite_data`

20.40.2.85 DISABLE_OAM_DMA `#define DISABLE_OAM_DMA _shadow_OAM_base = 0`

20.40.2.86 DISABLE_VBL_TRANSFER `#define DISABLE_VBL_TRANSFER DISABLE_OAM_DMA`
Disable OAM DMA copy each VBlank

20.40.2.87 ENABLE_OAM_DMA `#define ENABLE_OAM_DMA _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM
>> 8)`

20.40.2.88 ENABLE_VBL_TRANSFER `#define ENABLE_VBL_TRANSFER ENABLE_OAM_DMA`
Enable OAM DMA copy each VBlank and set it to transfer default shadow_OAM array

20.40.2.89 MAX_HARDWARE_SPRITES `#define MAX_HARDWARE_SPRITES 40`
Amount of hardware sprites in OAM

20.40.2.90 fill_rect `#define fill_rect fill_bkg_rect`

20.40.3 Typedef Documentation

20.40.3.1 int_handler `typedef void(* int_handler) (void) NONBANKED`
Interrupt handlers

20.40.3.2 OAM_item_t `typedef struct OAM_item_t OAM_item_t`
Sprite Attributes structure

Parameters

<i>x</i>	X Coordinate of the sprite on screen
<i>y</i>	Y Coordinate of the sprite on screen
<i>tile</i>	Sprite tile number (see set_sprite_tile)
<i>prop</i>	OAM Property Flags (see set_sprite_prop)

20.40.4 Function Documentation

20.40.4.1 remove_VBL() `void remove_VBL (
 int_handler h)`

The remove functions will remove any interrupt handler.
A handler of NULL will cause bad things to happen if the given interrupt is enabled.
Removes the VBL interrupt handler.

See also

[add_VBL\(\)](#)

Removes the VBL interrupt handler.

See also

[add_VBL\(\)](#)

20.40.4.2 remove_LCD() `void remove_LCD (`
`int_handler h)`

Removes the LCD interrupt handler.

See also

[add_LCD\(\)](#), [remove_VBL\(\)](#)

20.40.4.3 remove_TIM() `void remove_TIM (`
`int_handler h)`

Removes the TIM interrupt handler.

See also

[add_TIM\(\)](#), [remove_VBL\(\)](#)

20.40.4.4 remove_SIO() `void remove_SIO (`
`int_handler h)`

Removes the Serial Link / SIO interrupt handler.

See also

[add_SIO\(\)](#),
[remove_VBL\(\)](#)

The default SIO ISR gets installed automatically if any of the standard SIO calls are used. These calls include [add_SIO\(\)](#), [remove_SIO\(\)](#), [send_byte\(\)](#), [receive_byte\(\)](#).

The default SIO ISR cannot be removed once installed. Only secondary chained SIO ISRs (added with [add_SIO\(\)](#)) can be removed.

20.40.4.5 remove_JOY() `void remove_JOY (`
`int_handler h)`

Removes the JOY interrupt handler.

See also

[add_JOY\(\)](#), [remove_VBL\(\)](#)

20.40.4.6 add_VBL() `void add_VBL (`
`int_handler h)`

Adds a Vertical Blanking interrupt handler.

Parameters

<i>h</i>	The handler to be called whenever a V-blank interrupt occurs.
----------	---

Up to 4 handlers may be added, with the last added being called last.

Do not use the function definition attributes [CRITICAL](#) and [INTERRUPT](#) when declaring ISR functions added via [add_VBL\(\)](#) (or LCD, etc). Those attributes are only required when constructing a bare jump from the interrupt vector itself (such as with [ISR_VECTOR\(\)](#)).

ISR handlers added using [add_VBL\(\)](#)/etc are instead called via the GBDK ISR dispatcher which makes the extra function attributes unnecessary.

Note: The default GBDK VBL is installed automatically.

See also

[ISR_VECTOR\(\)](#)

Adds a V-blank interrupt handler.

20.40.4.7 add_LCD() `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](#).

Up to 3 handlers may be added, with the last added being called last.

There are various reasons for this interrupt to occur as described by the [STAT_REG](#) register (\$FF41). 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_REG](#) / [SCY_REG](#) registers (\$FF43/\$FF42) to perform special video effects.

Do not use the function definition attributes [CRITICAL](#) and [INTERRUPT](#) when declaring ISR functions added via [add_VBL\(\)](#) (or LCD, etc). Those attributes are only required when constructing a bare jump from the interrupt vector itself (such as with [ISR_VECTOR\(\)](#)).

ISR handlers added using [add_VBL\(\)](#)/etc are instead called via the GBDK ISR dispatcher which makes the extra function attributes unnecessary.

If this ISR is to be called once per each scanline then make sure that the time it takes to execute is less than the duration of a scanline.

See also

[add_VBL](#), [nowait_int_handler](#), [ISR_VECTOR\(\)](#)

Adds a LCD interrupt handler.

20.40.4.8 add_TIM() `void add_TIM (`
 [int_handler](#) *h*)

Adds a timer interrupt handler.

Can not be used together with [add_low_priority_TIM](#)

This interrupt occurs when the [TIMA_REG](#) register (\$FF05) changes from \$FF to \$00.

Up to 4 handlers may be added, with the last added being called last.

See also

[add_VBL](#)

[set_interrupts\(\)](#) with [TIM_IFLAG](#), [ISR_VECTOR\(\)](#)

20.40.4.9 add_low_priority_TIM() `void add_low_priority_TIM (`
 [int_handler](#) *h*)

Adds a timer interrupt handler, that could be interrupted by the other interrupts, as well as itself, if it runs too slow.

Can not be used together with [add_TIM](#)

This interrupt occurs when the [TIMA_REG](#) register (\$FF05) changes from \$FF to \$00.

Up to 4 handlers may be added, with the last added being called last.

See also

[add_VBL](#)

[set_interrupts\(\)](#) with [TIM_IFLAG](#), [ISR_VECTOR\(\)](#)

20.40.4.10 add_SIO() `void add_SIO (`
 `int_handler h)`

Adds a Serial Link transmit complete interrupt handler.

This interrupt occurs when a serial transfer has completed on the game link port.

Up to 4 handlers may be added, with the last added being called last.

See also

[send_byte](#), [receive_byte\(\)](#), [add_VBL\(\)](#)

[set_interrupts\(\)](#) with [SIO_IFLAG](#)

20.40.4.11 add_JOY() `void add_JOY (`
 `int_handler h)`

Adds a joystick button change interrupt handler.

This interrupt occurs on a transition of any of the keypad input lines from high to low. Due to the fact that keypad "bounce" is virtually always present, software should expect this interrupt to occur one or more times for every button press and one or more times for every button release.

Up to 4 handlers may be added, with the last added being called last.

See also

[joypad\(\)](#), [add_VBL\(\)](#)

20.40.4.12 nowait_int_handler() `void nowait_int_handler ()`

Interrupt handler chain terminator that does **not** wait for .STAT

You must add this handler last in every interrupt handler chain if you want to change the default interrupt handler behaviour that waits for LCD controller mode to become 1 or 0 before return from the interrupt.

Example:

```
CRITICAL {  
    add_SIO(nowait_int_handler); // Disable wait on VRAM state before returning from SIO interrupt  
}
```

See also

[wait_int_handler\(\)](#)

20.40.4.13 wait_int_handler() `void wait_int_handler ()`

Default Interrupt handler chain terminator that waits for

See also

[STAT_REG](#) and **only** returns at the BEGINNING of either Mode 0 or Mode 1.

Used by default at the end of interrupt chains to help prevent graphical glitches. The glitches are caused when an ISR interrupts a graphics operation in one mode but returns in a different mode for which that graphics operation is not allowed.

See also

[nowait_int_handler\(\)](#)

20.40.4.14 cancel_pending_interrupts() `uint8_t cancel_pending_interrupts () [inline]`

Cancel pending interrupts

20.40.4.15 mode() `void mode (`
 `uint8_t m)`

Set the current screen mode - one of M_* modes
Normally used by internal functions only.

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.40.4.16 get_mode() `uint8_t get_mode ()`

Returns the current mode

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.40.4.17 send_byte() `void send_byte ()`

Serial Link: Send the byte in [_io_out](#) out through the serial port
Make sure to enable interrupts for the Serial Link before trying to transfer data.

See also

[add_SIO\(\)](#), [remove_SIO\(\)](#)
[set_interrupts\(\)](#) with [SIO_IFLAG](#)

20.40.4.18 receive_byte() `void receive_byte ()`

Serial Link: Receive a byte from the serial port into [_io_in](#)
Make sure to enable interrupts for the Serial Link before trying to transfer data.

See also

[add_SIO\(\)](#), [remove_SIO\(\)](#)
[set_interrupts\(\)](#) with [SIO_IFLAG](#)

20.40.4.19 delay() `void delay (`
 `uint16_t d)`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

20.40.4.20 jypad() `uint8_t jypad ()`

Reads and returns the current state of the jypad. Follows Nintendo's guidelines for reading the pad. Return value is an OR of J_*

When testing for multiple different buttons, it's best to read the jypad state *once* into a variable and then test using that variable.

See also

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

20.40.4.21 waitpad() `uint8_t waitpad (`
 `uint8_t mask)`

Waits until at least one of the buttons given in mask are pressed.

Parameters

<i>mask</i>	Bitmask indicating which buttons to wait for
-------------	--

Normally only used for checking one key, but it will support many, even J_LEFT at the same time as J_RIGHT. :)
 Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

See also

[joypad](#)

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

20.40.4.22 waitpadup() `void waitpadup ()`

Waits for the directional pad and all buttons to be released.

Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

20.40.4.23 joypad_init() `uint8_t joypad_init (`
`uint8_t npads,`
`joypads_t * joypads)`

Initializes [joypads_t](#) structure for polling multiple joypads (for the GB and ones connected via SGB)

Parameters

<i>npads</i>	number of joypads requested (1, 2 or 4)
<i>joypads</i>	pointer to joypads_t structure to be initialized

Only required for [joypad_ex](#), not required for calls to regular [joypad\(\)](#)

Returns

number of joypads available

See also

[joypad_ex\(\)](#), [joypads_t](#)

20.40.4.24 joypad_ex() `void joypad_ex (`
`joypads_t * joypads)`

Polls all available joypads (for the GB and ones connected via SGB)

Parameters

<i>joypads</i>	pointer to joypads_t structure to be filled with joypad statuses, must be previously initialized with joypad_init()
----------------	---

See also

[joypad_init\(\)](#), [joypads_t](#)

20.40.4.25 enable_interrupts() `void enable_interrupts () [inline]`

Enables unmasked interrupts

Note

Use [CRITICAL](#) {...} instead for creating a block of code which should execute with interrupts temporarily turned off.

See also

[disable_interrupts](#), [set_interrupts](#), [CRITICAL](#)

20.40.4.26 disable_interrupts() `void disable_interrupts () [inline]`

Disables interrupts

Note

Use [CRITICAL](#) {...} instead for creating a block of code which should execute with interrupts temporarily turned off.

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](#), [set_interrupts](#), [CRITICAL](#)

20.40.4.27 set_interrupts() `void set_interrupts (
 uint8_t flags)`

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

Parameters

<i>flags</i>	A logical OR of *_IFLAGS
--------------	--------------------------

Note

: This disables and then re-enables interrupts so it must be used outside of a critical section.

See also

[enable_interrupts\(\)](#), [disable_interrupts\(\)](#)
[VBL_IFLAG](#), [LCD_IFLAG](#), [TIM_IFLAG](#), [SIO_IFLAG](#), [JOY_IFLAG](#)

20.40.4.28 reset() `void reset ()`

Performs a warm reset by reloading the CPU value then jumping to the start of crt0 (0x0150)

20.40.4.29 wait_vbl_done() `void wait_vbl_done ()`

HALTs the CPU and waits for the vertical blank interrupt (VBL) to finish.

This is often used in main loops to idle the CPU at low power until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediately.

20.40.4.30 display_off() `void display_off ()`

Turns the display off.

Waits until the VBL interrupt before turning the display off.

See also

[DISPLAY_ON](#)

20.40.4.31 refresh_OAM() `void refresh_OAM ()`

Copies data from shadow OAM to OAM

20.40.4.32 hiramcpy() `void hiramcpy (`

```
    uint8_t dst,
    const void * src,
    uint8_t n )
```

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

Parameters

<i>dst</i>	Offset in high ram (0xFF00 and above) to copy to.
<i>src</i>	Area to copy from
<i>n</i>	Number of bytes to copy.

20.40.4.33 set_vram_byte() `void set_vram_byte (`

```
    uint8_t * addr,
    uint8_t v )
```

Set byte in vram at given memory location

Parameters

<i>addr</i>	address to write to
<i>v</i>	value

20.40.4.34 get_vram_byte() `uint8_t get_vram_byte (`

```
    uint8_t * addr )
```

Get byte from vram at given memory location

Parameters

<i>addr</i>	address to read from
-------------	----------------------

Returns

read value

20.40.4.35 get_bkg_xy_addr() `uint8_t* get_bkg_xy_addr (`

```
    uint8_t x,
    uint8_t y )
```

Get address of X,Y tile of background map

20.40.4.36 set_2bpp_palette() `void set_2bpp_palette (`

```
    uint16_t palette ) [inline]
```

Sets palette for 2bpp color translation for GG/SMS, does nothing on GB

20.40.4.37 set_1bpp_colors_ex() void set_1bpp_colors_ex (
 uint8_t fgcolor,
 uint8_t bgcolor,
 uint8_t mode)

Sets the Foreground and Background colors used by the set_*_1bpp_*() functions

Parameters

<i>fgcolor</i>	Foreground color
<i>bgcolor</i>	Background color
<i>mode</i>	Draw Mode

See [set_1bpp_colors](#) for details.

20.40.4.38 set_1bpp_colors() void set_1bpp_colors (
 uint8_t fgcolor,
 uint8_t bgcolor) [inline]

Sets the Foreground and Background colors used by the set_*_1bpp_*() functions

Parameters

<i>fgcolor</i>	Foreground color to use
<i>bgcolor</i>	Background color to use

The default colors are:

- Foreground: DMG_BLACK
- Background: DMG_WHITE

Example:

```
// Use DMG_BLACK as the Foreground color and DMG_LITE_GRAY
// as the Background color when loading 1bpp tile data.
set_1bpp_colors(DMG_BLACK, DMG_LITE_GRAY);
```

See also

[DMG_BLACK](#), [DMG_DARK_GRAY](#), [DMG_LITE_GRAY](#), [DMG_WHITE](#)
[set_bkg_1bpp_data](#), [set_win_1bpp_data](#), [set_sprite_1bpp_data](#)

20.40.4.39 set_bkg_data() void set_bkg_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data)

Sets VRAM Tile Pattern data for the Background / Window

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source tile data

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- VBK_REG = [VBK_BANK_0](#) indicates the first bank
- VBK_REG = [VBK_BANK_1](#) indicates the second

See also

[set_win_data](#), [set_tile_data](#)

20.40.4.40 set_bkg_1bpp_data() void set_bkg_1bpp_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data)

Sets VRAM Tile Pattern data for the Background / Window using 1bpp source data

Parameters

<i>first_tile</i>	Index of the first Tile to write
<i>nb_tiles</i>	Number of Tiles to write
<i>data</i>	Pointer to (1bpp) source Tile Pattern data

Similar to [set_bkg_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel. For a given bit that represent a pixel:

- 0 will be expanded into the Background color
- 1 will be expanded into the Foreground color

See [set_1bpp_colors](#) for details about setting the Foreground and Background colors.

See also

[SHOW_BKG](#), [HIDE_BKG](#), [set_bkg_tiles](#)
[set_win_1bpp_data](#), [set_sprite_1bpp_data](#)

20.40.4.41 get_bkg_data() void get_bkg_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 uint8_t * data)

Copies from Background / Window VRAM Tile Pattern data into a buffer

Parameters

<i>first_tile</i>	Index of the first Tile to read from
<i>nb_tiles</i>	Number of Tiles to read
<i>data</i>	Pointer to destination buffer for Tile Pattern data

Copies **nb_tiles** tiles from VRAM starting at **first_tile**, Tile data is copied into **data**. Each Tile is 16 bytes, so the buffer pointed to by **data** should be at least **nb_tiles** x 16 bytes in size.

See also

[get_win_data](#), [get_data](#)

20.40.4.42 set_bkg_tiles() void set_bkg_tiles (
 uint8_t x,

```
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * tiles )
```

Sets a rectangular region of Background Tile Map.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data

Entries are copied from map at **tiles** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

Use [set_bkg_submap\(\)](#) instead when:

- Source map is wider than 32 tiles.
- Writing a width that does not match the source map width **and** more than one row high at a time.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

Note: Patterns 128-255 overlap with patterns 128-255 of the sprite Tile Pattern table.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- [VBK_REG](#) = [VBK_TILES](#) Tile Numbers are written
- [VBK_REG](#) = [VBK_ATTRIBUTES](#) Tile Attributes are written

GBC Tile 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: Flipped Vertically
- Bit 5 - Horizontal flip. Dictates which way up the tile is drawn horizontally.
0: Normal
1: Flipped Horizontally
- 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.

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_submap](#), [set_win_tiles](#), [set_tiles](#)

20.40.4.43 set_bkg_based_tiles() `void set_bkg_based_tiles (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * tiles,`
`uint8_t base_tile) [inline]`

Sets a rectangular region of Background Tile Map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_bkg_tiles\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_bkg_tiles](#) for more details

20.40.4.44 set_bkg_submap() `void set_bkg_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w) [inline]`

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 32 tiles.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: **x** & 0x1F and **y** & 0x1F). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: (**map_ptr** + **x** + (**y** * **map_width**)).

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: `map_ptr + 1 + (2 * map_width)`. Use this instead of [set_bkg_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See [set_bkg_tiles](#) for setting CGB attribute maps with [VBK_REG](#).

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_tiles](#), [set_win_submap](#), [set_tiles](#)

20.40.4.45 [set_bkg_based_submap\(\)](#) `void set_bkg_based_submap (`

```
uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * map,
uint8_t map_w,
uint8_t base_tile ) [inline]
```

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map_w</i>	Width of source tile map in tiles. Range 1 - 255
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_bkg_submap\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_bkg_submap](#) for more details

20.40.4.46 [get_bkg_tiles\(\)](#) `void get_bkg_tiles (`

```
uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
uint8_t * tiles )
```

Copies a rectangular region of Background Tile Map entries into a buffer.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
----------	---

Parameters

<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Background Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x** x **y** bytes in size.

See also

[get_win_tiles](#), [get_bkg_tile_xy](#), [get_tiles](#), [get_vram_byte](#)

20.40.4.47 set_bkg_tile_xy() `uint8_t* set_bkg_tile_xy (`
`uint8_t x,`
`uint8_t y,`
`uint8_t t)`

Set single tile t on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

20.40.4.48 get_bkg_tile_xy() `uint8_t get_bkg_tile_xy (`
`uint8_t x,`
`uint8_t y)`

Get single tile t on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate

Returns

returns tile index

20.40.4.49 move_bkg() `void move_bkg (`
`uint8_t x,`
`uint8_t y) [inline]`

Moves the Background Layer to the position specified in **x** and **y** in pixels.

Parameters

<i>x</i>	X axis screen coordinate for Left edge of the Background
<i>y</i>	Y axis screen coordinate for Top edge of the Background

0,0 is the top left corner of the GB screen. The Background Layer wraps around the screen, so when part of it goes off the screen it appears on the opposite side (factoring in the larger size of the Background Layer versus the screen size).

The background layer is always under the Window Layer.

See also

[SHOW_BKG](#), [HIDE_BKG](#)

20.40.4.50 scroll_bkg() `void scroll_bkg (`
 `int8_t x,`
 `int8_t y) [inline]`

Moves the Background relative to it's current position.

Parameters

<i>x</i>	Number of pixels to move the Background on the X axis Range: -128 - 127
<i>y</i>	Number of pixels to move the Background on the Y axis Range: -128 - 127

See also

[move_bkg](#)

20.40.4.51 get_win_xy_addr() `uint8_t* get_win_xy_addr (`
 `uint8_t x,`
 `uint8_t y)`

Get address of X,Y tile of window map

20.40.4.52 set_win_data() `void set_win_data (`
 `uint8_t first_tile,`
 `uint8_t nb_tiles,`
 `const uint8_t * data)`

Sets VRAM Tile Pattern data for the Window / Background

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data.

This is the same as [set_bkg_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

See also

[set_bkg_data](#)

[set_win_tiles](#), [set_bkg_data](#), [set_data](#)

[SHOW_WIN](#), [HIDE_WIN](#)

20.40.4.53 set_win_1bpp_data() void set_win_1bpp_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data)

Sets VRAM Tile Pattern data for the Window / Background using 1bpp source data

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (1bpp) source Tile Pattern data

This is the same as [set_bkg_1bpp_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

For a given bit that represent a pixel:

- 0 will be expanded into the Background color
- 1 will be expanded into the Foreground color

See [set_1bpp_colors](#) for details about setting the Foreground and Background colors.

See also

[set_bkg_data](#), [set_bkg_1bpp_data](#), [set_win_data](#), [set_1bpp_colors](#)
[set_bkg_1bpp_data](#), [set_sprite_1bpp_data](#)

20.40.4.54 get_win_data() void get_win_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 uint8_t * data)

Copies from Window / Background VRAM Tile Pattern data into a buffer

Parameters

<i>first_tile</i>	Index of the first Tile to read from
<i>nb_tiles</i>	Number of Tiles to read
<i>data</i>	Pointer to destination buffer for Tile Pattern Data

This is the same as [get_bkg_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

See also

[get_bkg_data](#), [get_data](#)

20.40.4.55 set_win_tiles() void set_win_tiles (
 uint8_t x,
 uint8_t y,
 uint8_t w,
 uint8_t h,
 const uint8_t * tiles)

Sets a rectangular region of the Window Tile Map.

Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data

Entries are copied from map at **tiles** to the Window Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

Use [set_win_submap\(\)](#) instead when:

- Source map is wider than 32 tiles.
- Writing a width that does not match the source map width **and** more than one row high at a time.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

Note: Patterns 128-255 overlap with patterns 128-255 of the sprite Tile Pattern table.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- [VBK_REG](#) = [VBK_TILES](#) Tile Numbers are written
- [VBK_REG](#) = [VBK_ATTRIBUTES](#) Tile Attributes are written

For more details about GBC Tile Attributes see [set_bkg_tiles](#).

See also

[SHOW_WIN](#), [HIDE_WIN](#), [set_win_submap](#), [set_bkg_tiles](#), [set_bkg_data](#), [set_tiles](#)

20.40.4.56 set_win_based_tiles() `void set_win_based_tiles (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * tiles,`
`uint8_t base_tile) [inline]`

Sets a rectangular region of the Window Tile Map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_win_tiles\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_win_tiles](#) for more details

20.40.4.57 set_win_submap() `void set_win_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w) [inline]`

Sets a rectangular area of the Window Tile Map using a sub-region from a source tile map.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Window Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: $x \& 0x1F$ and $y \& 0x1F$). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: $(map_ptr + x + (y * map_width))$.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: $map_ptr + 1 + (2 * map_width)$

Use this instead of [set_win_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- [VBK_REG](#) = [VBK_TILES](#) Tile Numbers are written
- [VBK_REG](#) = [VBK_ATTRIBUTES](#) Tile Attributes are written

See [set_bkg_tiles](#) for details about CGB attribute maps with [VBK_REG](#).

See also

[SHOW_WIN](#), [HIDE_WIN](#), [set_win_tiles](#), [set_bkg_submap](#), [set_bkg_tiles](#), [set_bkg_data](#), [set_tiles](#)

20.40.4.58 set_win_based_submap() `void set_win_based_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w,`
`uint8_t base_tile) [inline]`

Sets a rectangular area of the Window Tile Map using a sub-region from a source tile map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map_w</i>	Width of source tile map in tiles. Range 1 - 255
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_win_submap\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_win_submap](#) for more details

20.40.4.59 get_win_tiles() `void get_win_tiles (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t * tiles)`

Copies a rectangular region of Window Tile Map entries into a buffer.

Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Window Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x x y** bytes in size.

See also

[get_bkg_tiles](#), [get_bkg_tile_xy](#), [get_tiles](#), [get_vram_byte](#)

20.40.4.60 set_win_tile_xy() `uint8_t* set_win_tile_xy (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t t)`

Set single tile **t** on window layer at **x,y**

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

20.40.4.61 [get_win_tile_xy\(\)](#) `uint8_t get_win_tile_xy (`
`uint8_t x,`
`uint8_t y)`

Get single tile t on window layer at x,y

Parameters

<code>x</code>	X-coordinate
<code>y</code>	Y-coordinate

Returns

returns the tile index

20.40.4.62 [move_win\(\)](#) `void move_win (`
`uint8_t x,`
`uint8_t y) [inline]`

Moves the Window to the **x**, **y** position on the screen.

Parameters

<code>x</code>	X coordinate for Left edge of the Window (actual displayed location will be X - 7)
<code>y</code>	Y coordinate for Top edge of the Window

7,0 is the top left corner of the screen in Window coordinates. The Window is locked to the bottom right corner. The Window is always over the Background layer.

See also

[SHOW_WIN](#), [HIDE_WIN](#)

20.40.4.63 [scroll_win\(\)](#) `void scroll_win (`
`int8_t x,`
`int8_t y) [inline]`

Move the Window relative to its current position.

Parameters

<code>x</code>	Number of pixels to move the window on the X axis Range: -128 - 127
<code>y</code>	Number of pixels to move the window on the Y axis Range: -128 - 127

See also

[move_win](#)

20.40.4.64 set_sprite_data() void set_sprite_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data)

Sets VRAM Tile Pattern data for Sprites

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- VBK_REG = [VBK_BANK_0](#) indicates the first bank
- VBK_REG = [VBK_BANK_1](#) indicates the second

20.40.4.65 set_sprite_1bpp_data() void set_sprite_1bpp_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data)

Sets VRAM Tile Pattern data for Sprites using 1bpp source data

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (1bpp) source Tile Pattern data

Similar to [set_sprite_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel.

For a given bit that represent a pixel:

- 0 will be expanded into the Background color
- 1 will be expanded into the Foreground color

See [set_1bpp_colors](#) for details about setting the Foreground and Background colors.

See also

[SHOW_SPRITES](#), [HIDE_SPRITES](#), [set_sprite_tile](#)
[set_bkg_1bpp_data](#), [set_win_1bpp_data](#)

20.40.4.66 get_sprite_data() void get_sprite_data (
 uint8_t first_tile,
 uint8_t nb_tiles,
 uint8_t * data)

Copies from Sprite VRAM Tile Pattern data into a buffer

Parameters

<i>first_tile</i>	Index of the first tile to read from
-------------------	--------------------------------------

Parameters

<i>nb_tiles</i>	Number of tiles to read
<i>data</i>	Pointer to destination buffer for Tile Pattern data

Copies **nb_tiles** tiles from VRAM starting at **first_tile**, tile data is copied into **data**.

Each Tile is 16 bytes, so the buffer pointed to by **data** should be at least **nb_tiles** x 16 bytes in size.

20.40.4.67 SET_SHADOW_OAM_ADDRESS() `void SET_SHADOW_OAM_ADDRESS (void * address) [inline]`

Enable OAM DMA copy each VBlank and set it to transfer any 256-byte aligned array

20.40.4.68 set_sprite_tile() `void set_sprite_tile (uint8_t nb, uint8_t tile) [inline]`

Sets sprite number **nb** in the OAM to display tile number **tile**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>tile</i>	Selects a tile (0 - 255) from memory at 8000h - 8FFFh In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the OAM Attribute Flag (see set_sprite_prop)

In 8x16 mode:

- The sprite will also display the next tile (**tile** + 1) directly below (y + 8) the first tile.
- The lower bit of the tile number is ignored: the upper 8x8 tile is (**tile** & 0xFE), and the lower 8x8 tile is (**tile** | 0x01).
- See: [SPRITES_8x16](#)

20.40.4.69 get_sprite_tile() `uint8_t get_sprite_tile (uint8_t nb) [inline]`

Returns the tile number of sprite number **nb** in the OAM.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[set_sprite_tile](#) for more details

20.40.4.70 set_sprite_prop() `void set_sprite_prop (uint8_t nb, uint8_t prop) [inline]`

Sets the OAM Property Flags of sprite number **nb** to those defined in **prop**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>prop</i>	Property setting (see bitfield description)

The bits in **prop** represent:

- Bit 7 - Priority flag. When this is set the sprites appear behind the background and window layer.
0: infront
1: behind
- Bit 6 - Vertical flip. Dictates which way up the sprite is drawn vertically.
0: normal
1:upside down
- Bit 5 - Horizontal flip. Dictates which way up the sprite is drawn horizontally.
0: normal
1:back to front
- Bit 4 - DMG/Non-CGB Mode 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.

20.40.4.71 get_sprite_prop() `uint8_t get_sprite_prop (`
`uint8_t nb) [inline]`

Returns the OAM Property Flags of sprite number **nb**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[set_sprite_prop](#) for property bitfield settings

20.40.4.72 move_sprite() `void move_sprite (`
`uint8_t nb,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves sprite number **nb** to the **x, y** position on the screen.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	X Position. Specifies the sprites horizontal position on the screen (minus 8). An offscreen value (X=0 or X>=168) hides the sprite, but the sprite still affects the priority ordering - a better way to hide a sprite is to set its Y-coordinate offscreen.
<i>y</i>	Y Position. Specifies the sprites vertical position on the screen (minus 16). An offscreen value (for example, Y=0 or Y>=160) hides the sprite.

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

20.40.4.73 scroll_sprite() `void scroll_sprite (`
 `uint8_t nb,`
 `int8_t x,`
 `int8_t y) [inline]`

Moves sprite number **nb** relative to its current position.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	Number of pixels to move the sprite on the X axis Range: -128 - 127
<i>y</i>	Number of pixels to move the sprite on the Y axis Range: -128 - 127

See also

[move_sprite](#) for more details about the X and Y position

20.40.4.74 hide_sprite() `void hide_sprite (`
 `uint8_t nb) [inline]`

Hides sprite number **nb** by moving it to zero position by Y.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[hide_sprites_range](#), [HIDE_SPRITES](#)

20.40.4.75 set_data() `void set_data (`
 `uint8_t * vram_addr,`
 `const uint8_t * data,`
 `uint16_t len)`

Copies arbitrary data to an address in VRAM without taking into account the state of LCDC bits 3 or 4.

Parameters

<i>vram_addr</i>	Pointer to destination VRAM Address
<i>data</i>	Pointer to source buffer
<i>len</i>	Number of bytes to copy

Copies **len** bytes from a buffer at **data** to VRAM starting at **vram_addr**.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank
- `VBK_REG = VBK_BANK_1` indicates the second

See also

[set_bkg_data](#), [set_win_data](#), [set_bkg_tiles](#), [set_win_tiles](#), [set_tile_data](#), [set_tiles](#)

20.40.4.76 `get_data()` `void get_data (`
 `uint8_t * data,`
 `uint8_t * vram_addr,`
 `uint16_t len)`

Copies arbitrary data from an address in VRAM into a buffer without taking into account the state of LCDC bits 3 or 4.

Parameters

<i>vram_addr</i>	Pointer to source VRAM Address
<i>data</i>	Pointer to destination buffer
<i>len</i>	Number of bytes to copy

Copies **len** bytes from VRAM starting at **vram_addr** into a buffer at **data**.

GBC only: **VBK_REG** determines which bank of Background tile patterns are written to.

- **VBK_REG** = **VBK_BANK_0** indicates the first bank
- **VBK_REG** = **VBK_BANK_1** indicates the second

See also

[get_bkg_data](#), [get_win_data](#), [get_bkg_tiles](#), [get_win_tiles](#), [get_tiles](#)

20.40.4.77 `vmemcpy()` `void vmemcpy (`
 `uint8_t * dest,`
 `uint8_t * sour,`
 `uint16_t len)`

Copies arbitrary data from an address in VRAM into a buffer

Parameters

<i>dest</i>	Pointer to destination buffer (may be in VRAM)
<i>sour</i>	Pointer to source buffer (may be in VRAM)
<i>len</i>	Number of bytes to copy

Copies **len** bytes from or to VRAM starting at **sour** into a buffer or to VRAM at **dest**.

GBC only: **VBK_REG** determines which bank of Background tile patterns are written to.

- **VBK_REG** = **VBK_BANK_0** indicates the first bank
- **VBK_REG** = **VBK_BANK_1** indicates the second

20.40.4.78 `set_tiles()` `void set_tiles (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t * vram_addr,`
 `const uint8_t * tiles)`

Sets a rectangular region of Tile Map entries at a given VRAM Address without taking into account the state of LCDC bit 3.

Parameters

<i>x</i>	X Start position in Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>vram_addr</i>	Pointer to destination VRAM Address
<i>tiles</i>	Pointer to source Tile Map data

Entries are copied from **tiles** to Tile Map at address *vram_addr* starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

One byte per source tile map entry.

There are two 32x32 Tile Maps in VRAM at addresses 9800h-9BFFh and 9C00h-9FFFh.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- `VBK_REG = VBK_TILES` Tile Numbers are written
- `VBK_REG = VBK_ATTRIBUTES` Tile Attributes are written

See also

[set_bkg_tiles](#), [set_win_tiles](#)

```
20.40.4.79 set_tile_data() void set_tile_data (
    uint8_t first_tile,
    uint8_t nb_tiles,
    const uint8_t * data,
    uint8_t base )
```

Sets VRAM Tile Pattern data starting from given base address without taking into account the state of LCDC bit 4.

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data.
<i>base</i>	MSB of the destination address in VRAM (usually 0x80 or 0x90 which gives 0x8000 or 0x9000)

See also

[set_bkg_data](#), [set_win_data](#), [set_data](#)

```
20.40.4.80 get_tiles() void get_tiles (
    uint8_t x,
    uint8_t y,
    uint8_t w,
    uint8_t h,
    uint8_t * vram_addr,
    uint8_t * tiles )
```

Copies a rectangular region of Tile Map entries from a given VRAM Address into a buffer without taking into account the state of LCDC bit 3.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
----------	---

Parameters

<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>vram_addr</i>	Pointer to source VRAM Address
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Background Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

There are two 32x32 Tile Maps in VRAM at addresses 9800h - 9BFFh and 9C00h - 9FFFh.

The buffer pointed to by **tiles** should be at least **x** x **y** bytes in size.

See also

[get_bkg_tiles](#), [get_win_tiles](#)

20.40.4.81 set_native_tile_data() `void set_native_tile_data (`
 `uint16_t first_tile,`
 `uint8_t nb_tiles,`
 `const uint8_t * data) [inline]`

Sets VRAM Tile Pattern data in the native format

Parameters

<i>first_tile</i>	Index of the first tile to write (0 - 511)
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to source Tile Pattern data.

When *first_tile* is larger than 256 on the GB/AP, it will write to sprite data instead of background data.

The bit depth of the source Tile Pattern data depends on which console is being used:

- Game Boy/Analogue Pocket: loads 2bpp tiles data
- SMS/GG: loads 4bpp tile data

20.40.4.82 init_win() `void init_win (`
 `uint8_t c)`

Initializes the entire Window Tile Map with Tile Number **c**

Parameters

<i>c</i>	Tile number to fill with
----------	--------------------------

Note: This function avoids writes during modes 2 & 3

20.40.4.83 init_bkg() `void init_bkg (`
 `uint8_t c)`

Initializes the entire Background Tile Map with Tile Number **c**

Parameters

<i>c</i>	Tile number to fill with
----------	--------------------------

Note: This function avoids writes during modes 2 & 3

20.40.4.84 vmemset() `void vmemset (`
 `void * s,`
 `uint8_t c,`
 `size_t n)`

Fills the VRAM memory region **s** of size **n** with Tile Number **c**

Parameters

<i>s</i>	Start address in VRAM
<i>c</i>	Tile number to fill with
<i>n</i>	Size of memory region (in bytes) to fill

Note: This function avoids writes during modes 2 & 3

20.40.4.85 fill_bkg_rect() `void fill_bkg_rect (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t tile)`

Fills a rectangular region of Tile Map entries for the Background layer with tile.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 0 - 31
<i>h</i>	Height of area to set in tiles. Range 0 - 31
<i>tile</i>	Fill value

20.40.4.86 fill_win_rect() `void fill_win_rect (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t tile)`

Fills a rectangular region of Tile Map entries for the Window layer with tile.

Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 0 - 31
<i>h</i>	Height of area to set in tiles. Range 0 - 31
<i>tile</i>	Fill value

20.40.5 Variable Documentation

20.40.5.1 `c` `void c`

20.40.5.2 `_cpu` `uint8_t _cpu`
GB CPU type

See also

[DMG_TYPE](#), [MGB_TYPE](#), [CGB_TYPE](#), [cpu_fast\(\)](#), [cpu_slow\(\)](#), [_is_GBA](#)

20.40.5.3 `_is_GBA` `uint8_t _is_GBA`
GBA detection

See also

[GBA_DETECTED](#), [GBA_NOT_DETECTED](#), [_cpu](#)

20.40.5.4 `sys_time` `volatile uint16_t sys_time`
Global Time Counter in VBL periods (60Hz)
Increments once per Frame
Will wrap around every ~18 minutes (unsigned 16 bits = 65535 / 60 / 60 = 18.2)

20.40.5.5 `_io_status` `volatile uint8_t _io_status`
Serial Link: Current IO Status. An OR of `IO_*`

20.40.5.6 `_io_in` `volatile uint8_t _io_in`
Serial Link: Byte just read after calling [receive_byte\(\)](#)

20.40.5.7 `_io_out` `volatile uint8_t _io_out`
Serial Link: Write byte to send here before calling [send_byte\(\)](#)

20.40.5.8 `_current_bank` `__REG _current_bank`
Tracks current active ROM bank
The active bank number is not tracked by `_current_bank` when [SWITCH_ROM_MBC5_8M](#) is used.
This variable is updated automatically when you call [SWITCH_ROM_MBC1](#) or [SWITCH_ROM_MBC5](#), [SWITCH_ROM\(\)](#), or call a BANKED function.

See also

[SWITCH_ROM_MBC1\(\)](#), [SWITCH_ROM_MBC5\(\)](#), [SWITCH_ROM\(\)](#)

20.40.5.9 `l` `uint8_t l`

Initial value:

```
{  
    __asm__("ei")  
}
```

20.40.5.10 `h` `uint8_t h`

20.40.5.11 `b` `void b`

20.40.5.12 `d` `void d`

20.40.5.13 `e` `void e`

20.40.5.14 `_current_1bpp_colors` `uint16_t _current_1bpp_colors`

20.40.5.15 `_map_tile_offset` `uint8_t _map_tile_offset`

20.40.5.16 `_submap_tile_offset` `uint8_t _submap_tile_offset`

20.40.5.17 `shadow_OAM` `volatile struct OAM_item_t shadow_OAM[]`
Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

20.40.5.18 `_shadow_OAM_base` `__REG _shadow_OAM_base`
MSB of shadow_OAM address is used by OAM DMA copying routine

20.41 gbdk-lib/include/gb/gbdecompress.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Functions

- `uint16_t gb_decompress` (`const uint8_t *sour`, `uint8_t *dest`) `OLDCALL PRESERVES_REGS(b`
- `void gb_decompress_bkg_data` (`uint8_t first_tile`, `const uint8_t *sour`) `OLDCALL PRESERVES_REGS(b`
- `void gb_decompress_win_data` (`uint8_t first_tile`, `const uint8_t *sour`) `OLDCALL PRESERVES_REGS(b`
- `void gb_decompress_sprite_data` (`uint8_t first_tile`, `const uint8_t *sour`) `OLDCALL PRESERVES_REGS(b`

Variables

- `uint16_t c`

20.41.1 Detailed Description

GB-Compress decompressor Compatible with the compression used in GBTD

See also

`utility_gbcompress` "gbcompress"

GB-Compress decompressor Compatible with the compression used in GBTD

20.41.2 Function Documentation

20.41.2.1 `gb_decompress()` `uint16_t gb_decompress` (
 `const uint8_t * sour`,
 `uint8_t * dest`)
gb-decompress data from sour into dest

Parameters

<i>sour</i>	Pointer to source gb-compressed data
<i>dest</i>	Pointer to destination buffer/address

Will decompress **all** of it's data to destination without stopping until the end of compressed data is reached. It is not possible to set a limit, so ensure the destination buffer has sufficient space to avoid an overflow.

See also

[gb_decompress_bkg_data](#), [gb_decompress_win_data](#), [gb_decompress_sprite_data](#), [rle_decompress](#)

20.41.2.2 gb_decompress_bkg_data() `void gb_decompress_bkg_data (`
`uint8_t first_tile,`
`const uint8_t * sour)`

gb-decompress background tiles into VRAM

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>sour</i>	Pointer to (gb-compressed 2 bpp) source Tile Pattern data.

Note: This function avoids writes during modes 2 & 3

Will decompress **all** of it's data to destination without stopping until the end of compressed data is reached. It is not possible to set a limit, so ensure the destination buffer has sufficient space to avoid an overflow.

See also

[gb_decompress_bkg_data](#), [gb_decompress_win_data](#), [gb_decompress_sprite_data](#)

20.41.2.3 gb_decompress_win_data() `void gb_decompress_win_data (`
`uint8_t first_tile,`
`const uint8_t * sour)`

gb-decompress window tiles into VRAM

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>sour</i>	Pointer to (gb-compressed 2 bpp) source Tile Pattern data.

This is the same as [gb_decompress_bkg_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

Note: This function avoids writes during modes 2 & 3

Will decompress **all** of it's data to destination without stopping until the end of compressed data is reached. It is not possible to set a limit, so ensure the destination buffer has sufficient space to avoid an overflow.

See also

[gb_decompress](#), [gb_decompress_bkg_data](#), [gb_decompress_sprite_data](#)

20.41.2.4 gb_decompress_sprite_data() `void gb_decompress_sprite_data (`
`uint8_t first_tile,`
`const uint8_t * sour)`

gb-decompress sprite tiles into VRAM

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>sour</i>	Pointer to source compressed data

Note: This function avoids writes during modes 2 & 3

Will decompress **all** of it's data to destination without stopping until the end of compressed data is reached. It is not possible to set a limit, so ensure the destination buffer has sufficient space to avoid an overflow.

See also

[gb_decompress](#), [gb_decompress_bkg_data](#), [gb_decompress_win_data](#)

20.41.3 Variable Documentation

20.41.3.1 `c` `void c`

20.42 gbdk-lib/include/gbdk/gbdecompress.h File Reference

```
#include <gb/gbdecompress.h>
```

20.43 gbdk-lib/include/sms/gbdecompress.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Functions

- [uint16_t gb_decompress](#) (const [uint8_t](#) *sour, [uint8_t](#) *dest) [Z88DK_CALLEE PRESERVES_REGS](#)(b

Variables

- [uint16_t c](#)

20.43.1 Function Documentation

20.43.1.1 `gb_decompress()` `uint16_t gb_decompress (` `const uint8_t * sour,` `uint8_t * dest)`

gb-decompress data from sour into dest

Parameters

<i>sour</i>	Pointer to source gb-compressed data
<i>dest</i>	Pointer to destination buffer/address

Returns

Return value is number of bytes decompressed

See also

[gb_decompress_bkg_data](#), [gb_decompress_win_data](#), [gb_decompress_sprite_data](#)

20.43.2 Variable Documentation

20.43.2.1 c uint16_t c

20.44 gbdk-lib/include/gb/hardware.h File Reference

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

Macros

- #define [__BYTES](#) extern [UBYTE](#)
- #define [__BYTE_REG](#) extern volatile [UBYTE](#)
- #define [__REG](#) extern volatile [SFR](#)
- #define [rP1](#) [P1_REG](#)
- #define [P1F_5](#) 0b00100000
- #define [P1F_4](#) 0b00010000
- #define [P1F_3](#) 0b00001000
- #define [P1F_2](#) 0b00000100
- #define [P1F_1](#) 0b00000010
- #define [P1F_0](#) 0b00000001
- #define [P1F_GET_DPAD](#) [P1F_5](#)
- #define [P1F_GET_BTN](#) [P1F_4](#)
- #define [P1F_GET_NONE](#) ([P1F_4](#) | [P1F_5](#))
- #define [rSB](#) [SB_REG](#)
- #define [rSC](#) [SC_REG](#)
- #define [rDIV](#) [DIV_REG](#)
- #define [rTIMA](#) [TIMA_REG](#)
- #define [rTMA](#) [TMA_REG](#)
- #define [rTAC](#) [TAC_REG](#)
- #define [TACF_START](#) 0b00000100
- #define [TACF_STOP](#) 0b00000000
- #define [TACF_4KHZ](#) 0b00000000
- #define [TACF_16KHZ](#) 0b00000011
- #define [TACF_65KHZ](#) 0b00000010
- #define [TACF_262KHZ](#) 0b00000001
- #define [SIOF_CLOCK_EXT](#) 0b00000000
- #define [SIOF_CLOCK_INT](#) 0b00000001
- #define [SIOF_SPEED_1X](#) 0b00000000
- #define [SIOF_SPEED_32X](#) 0b00000010
- #define [SIOF_XFER_START](#) 0b10000000
- #define [SIOF_B_CLOCK](#) 0
- #define [SIOF_B_SPEED](#) 1
- #define [SIOF_B_XFER_START](#) 7
- #define [rIF](#) [IF_REG](#)
- #define [rAUD1SWEEP](#) [NR10_REG](#)
- #define [AUD1SWEEP_UP](#) 0b00000000
- #define [AUD1SWEEP_DOWN](#) 0b00001000
- #define [AUD1SWEEP_TIME](#)(x) ((x) << 4)
- #define [AUD1SWEEP_LENGTH](#)(x) (x)

- `#define rAUD1LEN NR11_REG`
- `#define rAUD1ENV NR12_REG`
- `#define rAUD1LOW NR13_REG`
- `#define rAUD1HIGH NR14_REG`
- `#define rAUD2LEN NR21_REG`
- `#define rAUD2ENV NR22_REG`
- `#define rAUD2LOW NR23_REG`
- `#define rAUD2HIGH NR24_REG`
- `#define rAUD3ENA NR30_REG`
- `#define rAUD3LEN NR31_REG`
- `#define rAUD3LEVEL NR32_REG`
- `#define rAUD3LOW NR33_REG`
- `#define rAUD3HIGH NR34_REG`
- `#define rAUD4LEN NR41_REG`
- `#define rAUD4ENV NR42_REG`
- `#define rAUD4POLY NR43_REG`
- `#define AUD4POLY_WIDTH_15BIT 0x00`
- `#define AUD4POLY_WIDTH_7BIT 0x08`
- `#define rAUD4GO NR44_REG`
- `#define rAUDVOL NR50_REG`
- `#define AUDVOL_VOL_LEFT(x) ((x) << 4)`
- `#define AUDVOL_VOL_RIGHT(x) ((x))`
- `#define AUDVOL_VIN_LEFT 0b10000000`
- `#define AUDVOL_VIN_RIGHT 0b00001000`
- `#define rAUDTERM NR51_REG`
- `#define AUDTERM_4_LEFT 0b10000000`
- `#define AUDTERM_3_LEFT 0b01000000`
- `#define AUDTERM_2_LEFT 0b00100000`
- `#define AUDTERM_1_LEFT 0b00010000`
- `#define AUDTERM_4_RIGHT 0b00001000`
- `#define AUDTERM_3_RIGHT 0b00000100`
- `#define AUDTERM_2_RIGHT 0b00000010`
- `#define AUDTERM_1_RIGHT 0b00000001`
- `#define rAUDENA NR52_REG`
- `#define AUDENA_ON 0b10000000`
- `#define AUDENA_OFF 0b00000000`
- `#define rLDCDC LDCDC_REG`
- `#define LCDCF_OFF 0b00000000`
- `#define LCDCF_ON 0b10000000`
- `#define LCDCF_WIN9800 0b00000000`
- `#define LCDCF_WIN9C00 0b01000000`
- `#define LCDCF_WINOFF 0b00000000`
- `#define LCDCF_WINON 0b00100000`
- `#define LCDCF_BG8800 0b00000000`
- `#define LCDCF_BG8000 0b00010000`
- `#define LCDCF_BG9800 0b00000000`
- `#define LCDCF_BG9C00 0b00001000`
- `#define LCDCF_OBJ8 0b00000000`
- `#define LCDCF_OBJ16 0b00000100`
- `#define LCDCF_OBJOFF 0b00000000`
- `#define LCDCF_OBJON 0b00000010`
- `#define LCDCF_BG0FF 0b00000000`
- `#define LCDCF_BGON 0b00000001`
- `#define LCDCF_B_ON 7`
- `#define LCDCF_B_WIN9C00 6`

- `#define LCDCF_B_WINON 5`
- `#define LCDCF_B_BG8000 4`
- `#define LCDCF_B_BG9C00 3`
- `#define LCDCF_B_OBJ16 2`
- `#define LCDCF_B_OBJON 1`
- `#define LCDCF_B_BGON 0`
- `#define rSTAT STAT_REG`
- `#define STATF_LYC 0b01000000`
- `#define STATF_MODE10 0b00100000`
- `#define STATF_MODE01 0b00010000`
- `#define STATF_MODE00 0b00001000`
- `#define STATF_LYCF 0b00000100`
- `#define STATF_HBL 0b00000000`
- `#define STATF_VBL 0b00000001`
- `#define STATF_OAM 0b00000010`
- `#define STATF_LCD 0b00000011`
- `#define STATF_BUSY 0b00000010`
- `#define STATF_B_LYC 6`
- `#define STATF_B_MODE10 5`
- `#define STATF_B_MODE01 4`
- `#define STATF_B_MODE00 3`
- `#define STATF_B_LYCF 2`
- `#define STATF_B_VBL 0`
- `#define STATF_B_OAM 1`
- `#define STATF_B_BUSY 1`
- `#define rSCY`
- `#define rSCX SCX_REG`
- `#define rLY LY_REG`
- `#define rLYC LYC_REG`
- `#define rDMA DMA_REG`
- `#define rBGP BGP_REG`
- `#define rOBP0 OBP0_REG`
- `#define rOBP1 OBP1_REG`
- `#define rWY WY_REG`
- `#define rWX WX_REG`
- `#define rKEY1 KEY1_REG`
- `#define rSPD KEY1_REG`
- `#define KEY1F_DBLSPED 0b10000000`
- `#define KEY1F_PREPARE 0b00000001`
- `#define rVBK VBK_REG`
- `#define VBK_BANK_0 0`
- `#define VBK_TILES 0`
- `#define VBK_BANK_1 1`
- `#define VBK_ATTRIBUTES 1`
- `#define BKGF_PRI 0b10000000`
- `#define BKGF_YFLIP 0b01000000`
- `#define BKGF_XFLIP 0b00100000`
- `#define BKGF_BANK0 0b00000000`
- `#define BKGF_BANK1 0b00001000`
- `#define BKGF_CGB_PAL0 0b00000000`
- `#define BKGF_CGB_PAL1 0b00000001`
- `#define BKGF_CGB_PAL2 0b00000010`
- `#define BKGF_CGB_PAL3 0b00000011`
- `#define BKGF_CGB_PAL4 0b00000100`
- `#define BKGF_CGB_PAL5 0b00000101`

- #define BKGf_CGB_PAL6 0b00000110
- #define BKGf_CGB_PAL7 0b00000111
- #define rHDMA1 HDMA1_REG
- #define rHDMA2 HDMA2_REG
- #define rHDMA3 HDMA3_REG
- #define rHDMA4 HDMA4_REG
- #define rHDMA5 HDMA5_REG
- #define HDMA5F_MODE_GP 0b00000000
- #define HDMA5F_MODE_HBL 0b10000000
- #define HDMA5F_BUSY 0b10000000
- #define rRP RP_REG
- #define RPF_ENREAD 0b11000000
- #define RPF_DATAIN 0b00000010
- #define RPF_WRITE_HI 0b00000001
- #define RPF_WRITE_LO 0b00000000
- #define rBCPS BCPS_REG
- #define BCPSF_AUTOINC 0b10000000
- #define rBCPD BCPD_REG
- #define rOCPS OCPS_REG
- #define OCPSF_AUTOINC 0b10000000
- #define rOCPD OCPD_REG
- #define rSVBK SVBK_REG
- #define rSMBK SVBK_REG
- #define rPCM12 PCM12_REG
- #define rPCM34 PCM34_REG
- #define rIE IE_REG
- #define IEF_HILO 0b00010000
- #define IEF_SERIAL 0b00001000
- #define IEF_TIMER 0b00000100
- #define IEF_STAT 0b00000010
- #define IEF_VBLANK 0b00000001
- #define AUDLEN_DUTY_12_5 0b00000000
- #define AUDLEN_DUTY_25 0b01000000
- #define AUDLEN_DUTY_50 0b10000000
- #define AUDLEN_DUTY_75 0b11000000
- #define AUDLEN_LENGTH(x) (x)
- #define AUDENV_VOL(x) ((x) << 4)
- #define AUDENV_UP 0b00001000
- #define AUDENV_DOWN 0b00000000
- #define AUDENV_LENGTH(x) (x)
- #define AUDHIGH_RESTART 0b10000000
- #define AUDHIGH_LENGTH_ON 0b01000000
- #define AUDHIGH_LENGTH_OFF 0b00000000
- #define OAMF_PRI 0b10000000
- #define OAMF_YFLIP 0b01000000
- #define OAMF_XFLIP 0b00100000
- #define OAMF_PAL0 0b00000000
- #define OAMF_PAL1 0b00010000
- #define OAMF_BANK0 0b00000000
- #define OAMF_BANK1 0b00001000
- #define OAMF_CGB_PAL0 0b00000000
- #define OAMF_CGB_PAL1 0b00000001
- #define OAMF_CGB_PAL2 0b00000010
- #define OAMF_CGB_PAL3 0b00000011
- #define OAMF_CGB_PAL4 0b00000100

- `#define OAMF_CGB_PAL5 0b00000101`
- `#define OAMF_CGB_PAL6 0b00000110`
- `#define OAMF_CGB_PAL7 0b00000111`
- `#define OAMF_PALMASK 0b00000111`
- `#define DEVICE_SCREEN_X_OFFSET 0`
- `#define DEVICE_SCREEN_Y_OFFSET 0`
- `#define DEVICE_SCREEN_WIDTH 20`
- `#define DEVICE_SCREEN_HEIGHT 18`
- `#define DEVICE_SCREEN_BUFFER_WIDTH 32`
- `#define DEVICE_SCREEN_BUFFER_HEIGHT 32`
- `#define DEVICE_SCREEN_MAP_ENTRY_SIZE 1`
- `#define DEVICE_SPRITE_PX_OFFSET_X 8`
- `#define DEVICE_SPRITE_PX_OFFSET_Y 16`
- `#define DEVICE_WINDOW_PX_OFFSET_X 7`
- `#define DEVICE_WINDOW_PX_OFFSET_Y 0`
- `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`
- `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

Variables

- `__BYTES_VRAM []`
- `__BYTES_VRAM8000 []`
- `__BYTES_VRAM8800 []`
- `__BYTES_VRAM9000 []`
- `__BYTES_SCRN0 []`
- `__BYTES_SCRN1 []`
- `__BYTES_SRAM []`
- `__BYTES_RAM []`
- `__BYTES_RAMBANK []`
- `__BYTES_OAMRAM []`
- `__BYTE_REG_IO []`
- `__BYTE_REG_AUD3WAVERAM []`
- `__BYTE_REG_HRAM []`
- `__BYTE_REG rRAMG`
- `__BYTE_REG rROMB0`
- `__BYTE_REG rROMB1`
- `__BYTE_REG rRAMB`
- `__REG P1_REG`
- `__REG SB_REG`
- `__REG SC_REG`
- `__REG DIV_REG`
- `__REG TIMA_REG`
- `__REG TMA_REG`
- `__REG TAC_REG`
- `__REG IF_REG`
- `__REG NR10_REG`
- `__REG NR11_REG`
- `__REG NR12_REG`
- `__REG NR13_REG`
- `__REG NR14_REG`
- `__REG NR21_REG`
- `__REG NR22_REG`
- `__REG NR23_REG`
- `__REG NR24_REG`
- `__REG NR30_REG`

- [__REG NR31_REG](#)
- [__REG NR32_REG](#)
- [__REG NR33_REG](#)
- [__REG NR34_REG](#)
- [__REG NR41_REG](#)
- [__REG NR42_REG](#)
- [__REG NR43_REG](#)
- [__REG NR44_REG](#)
- [__REG NR50_REG](#)
- [__REG NR51_REG](#)
- [__REG NR52_REG](#)
- [__BYTE_REG AUD3WAVE](#) [16]
- [__BYTE_REG PCM_SAMPLE](#) [16]
- [__REG LCDC_REG](#)
- [__REG STAT_REG](#)
- [__REG SCY_REG](#)
- [__REG SCX_REG](#)
- [__REG LY_REG](#)
- [__REG LYC_REG](#)
- [__REG DMA_REG](#)
- [__REG BGP_REG](#)
- [__REG OBP0_REG](#)
- [__REG OBP1_REG](#)
- [__REG WY_REG](#)
- [__REG WX_REG](#)
- [__REG KEY1_REG](#)
- [__REG VBK_REG](#)
- [__REG HDMA1_REG](#)
- [__REG HDMA2_REG](#)
- [__REG HDMA3_REG](#)
- [__REG HDMA4_REG](#)
- [__REG HDMA5_REG](#)
- [__REG RP_REG](#)
- [__REG BCPS_REG](#)
- [__REG BCPD_REG](#)
- [__REG OCPS_REG](#)
- [__REG OCPD_REG](#)
- [__REG SVBK_REG](#)
- [__REG PCM12_REG](#)
- [__REG PCM34_REG](#)
- [__REG IE_REG](#)

20.44.1 Detailed Description

Defines that let the GB's hardware registers be accessed from C.
See the [Pandocs](#) for more details on each register.

20.44.2 Macro Definition Documentation

20.44.2.1 [__BYTES](#) `#define __BYTES extern UBYTE`

20.44.2.2 [__BYTE_REG](#) `#define __BYTE_REG extern volatile UBYTE`

20.44.2.3 `__REG` `#define __REG extern volatile SFR`

20.44.2.4 `rP1` `#define rP1 P1_REG`

20.44.2.5 `P1F_5` `#define P1F_5 0b00100000`

20.44.2.6 `P1F_4` `#define P1F_4 0b00010000`

20.44.2.7 `P1F_3` `#define P1F_3 0b00001000`

20.44.2.8 `P1F_2` `#define P1F_2 0b00000100`

20.44.2.9 `P1F_1` `#define P1F_1 0b00000010`

20.44.2.10 `P1F_0` `#define P1F_0 0b00000001`

20.44.2.11 `P1F_GET_DPAD` `#define P1F_GET_DPAD P1F_5`

20.44.2.12 `P1F_GET_BTN` `#define P1F_GET_BTN P1F_4`

20.44.2.13 `P1F_GET_NONE` `#define P1F_GET_NONE (P1F_4 | P1F_5)`

20.44.2.14 `rSB` `#define rSB SB_REG`

20.44.2.15 `rSC` `#define rSC SC_REG`

20.44.2.16 `rDIV` `#define rDIV DIV_REG`

20.44.2.17 `rTIMA` `#define rTIMA TIMA_REG`

20.44.2.18 `rTMA` `#define rTMA TMA_REG`

20.44.2.19 `rTAC` `#define rTAC TAC_REG`

20.44.2.20 `TACF_START` `#define TACF_START 0b00000100`

20.44.2.21 TACF_STOP `#define TACF_STOP 0b00000000`

20.44.2.22 TACF_4KHZ `#define TACF_4KHZ 0b00000000`

20.44.2.23 TACF_16KHZ `#define TACF_16KHZ 0b00000011`

20.44.2.24 TACF_65KHZ `#define TACF_65KHZ 0b00000010`

20.44.2.25 TACF_262KHZ `#define TACF_262KHZ 0b00000001`

20.44.2.26 SIOF_CLOCK_EXT `#define SIOF_CLOCK_EXT 0b00000000`

Serial IO: Use External clock

20.44.2.27 SIOF_CLOCK_INT `#define SIOF_CLOCK_INT 0b00000001`

Serial IO: Use Internal clock

20.44.2.28 SIOF_SPEED_1X `#define SIOF_SPEED_1X 0b00000000`

Serial IO: If internal clock then 8KHz mode, 1KB/s (16KHz in CGB high-speed mode, 2KB/s)

20.44.2.29 SIOF_SPEED_32X `#define SIOF_SPEED_32X 0b00000010`

Serial IO: **CGB-Mode ONLY** If internal clock then 256KHz mode, 32KB/s (512KHz in CGB high-speed mode, 64KB/s)

20.44.2.30 SIOF_XFER_START `#define SIOF_XFER_START 0b10000000`

Serial IO: Start Transfer. Automatically cleared at the end of transfer

20.44.2.31 SIOF_B_CLOCK `#define SIOF_B_CLOCK 0`

20.44.2.32 SIOF_B_SPEED `#define SIOF_B_SPEED 1`

20.44.2.33 SIOF_B_XFER_START `#define SIOF_B_XFER_START 7`

20.44.2.34 rIF `#define rIF IF_REG`

20.44.2.35 rAUD1SWEEP `#define rAUD1SWEEP NR10_REG`

20.44.2.36 AUD1SWEEP_UP `#define AUD1SWEEP_UP 0b00000000`

20.44.2.37 AUD1SWEEP_DOWN `#define AUD1SWEEP_DOWN 0b00001000`

20.44.2.38 AUD1SWEEP_TIME `#define AUD1SWEEP_TIME(
x) ((x) << 4)`

20.44.2.39 AUD1SWEEP_LENGTH `#define AUD1SWEEP_LENGTH(
x) (x)`

20.44.2.40 rAUD1LEN `#define rAUD1LEN NR11_REG`

20.44.2.41 rAUD1ENV `#define rAUD1ENV NR12_REG`

20.44.2.42 rAUD1LOW `#define rAUD1LOW NR13_REG`

20.44.2.43 rAUD1HIGH `#define rAUD1HIGH NR14_REG`

20.44.2.44 rAUD2LEN `#define rAUD2LEN NR21_REG`

20.44.2.45 rAUD2ENV `#define rAUD2ENV NR22_REG`

20.44.2.46 rAUD2LOW `#define rAUD2LOW NR23_REG`

20.44.2.47 rAUD2HIGH `#define rAUD2HIGH NR24_REG`

20.44.2.48 rAUD3ENA `#define rAUD3ENA NR30_REG`

20.44.2.49 rAUD3LEN `#define rAUD3LEN NR31_REG`

20.44.2.50 rAUD3LEVEL `#define rAUD3LEVEL NR32_REG`

20.44.2.51 rAUD3LOW `#define rAUD3LOW NR33_REG`

20.44.2.52 rAUD3HIGH `#define rAUD3HIGH NR34_REG`

20.44.2.53 rAUD4LEN `#define rAUD4LEN NR41_REG`

20.44.2.54 rAUD4ENV `#define rAUD4ENV NR42_REG`

20.44.2.55 rAUD4POLY `#define rAUD4POLY NR43_REG`

20.44.2.56 AUD4POLY_WIDTH_15BIT `#define AUD4POLY_WIDTH_15BIT 0x00`

20.44.2.57 AUD4POLY_WIDTH_7BIT `#define AUD4POLY_WIDTH_7BIT 0x08`

20.44.2.58 rAUD4GO `#define rAUD4GO NR44_REG`

20.44.2.59 rAUDVOL `#define rAUDVOL NR50_REG`

20.44.2.60 AUDVOL_VOL_LEFT `#define AUDVOL_VOL_LEFT(
x) ((x) << 4)`

20.44.2.61 AUDVOL_VOL_RIGHT `#define AUDVOL_VOL_RIGHT(
x) ((x))`

20.44.2.62 AUDVOL_VIN_LEFT `#define AUDVOL_VIN_LEFT 0b10000000`

20.44.2.63 AUDVOL_VIN_RIGHT `#define AUDVOL_VIN_RIGHT 0b00001000`

20.44.2.64 rAUDTERM `#define rAUDTERM NR51_REG`

20.44.2.65 AUDTERM_4_LEFT `#define AUDTERM_4_LEFT 0b10000000`

20.44.2.66 AUDTERM_3_LEFT `#define AUDTERM_3_LEFT 0b01000000`

20.44.2.67 AUDTERM_2_LEFT `#define AUDTERM_2_LEFT 0b00100000`

20.44.2.68 AUDTERM_1_LEFT `#define AUDTERM_1_LEFT 0b00010000`

20.44.2.69 AUDTERM_4_RIGHT `#define AUDTERM_4_RIGHT 0b00001000`

20.44.2.70 AUDTERM_3_RIGHT `#define AUDTERM_3_RIGHT 0b00000100`

20.44.2.71 AUDTERM_2_RIGHT `#define AUDTERM_2_RIGHT 0b00000010`

20.44.2.72 AUDTERM_1_RIGHT `#define AUDTERM_1_RIGHT 0b00000001`

20.44.2.73 rAUDENA `#define rAUDENA NR52_REG`

20.44.2.74 AUDENA_ON `#define AUDENA_ON 0b10000000`

20.44.2.75 AUDENA_OFF `#define AUDENA_OFF 0b00000000`

20.44.2.76 rLCDC `#define rLCDC LCDC_REG`

20.44.2.77 LCDCF_OFF `#define LCDCF_OFF 0b00000000`
LCD Control: Off

20.44.2.78 LCDCF_ON `#define LCDCF_ON 0b10000000`
LCD Control: On

20.44.2.79 LCDCF_WIN9800 `#define LCDCF_WIN9800 0b00000000`
Window Tile Map: Use 9800 Region

20.44.2.80 LCDCF_WIN9C00 `#define LCDCF_WIN9C00 0b01000000`
Window Tile Map: Use 9C00 Region

20.44.2.81 LCDCF_WINOFF `#define LCDCF_WINOFF 0b00000000`
Window Display: Hidden

20.44.2.82 LCDCF_WINON `#define LCDCF_WINON 0b00100000`
Window Display: Visible

20.44.2.83 LCDCF_BG8800 `#define LCDCF_BG8800 0b00000000`
BG & Window Tile Data: Use 8800 Region

20.44.2.84 LCDCF_BG8000 `#define LCDCF_BG8000 0b00010000`
BG & Window Tile Data: Use 8000 Region

20.44.2.85 LCDCF_BG9800 `#define LCDCF_BG9800 0b00000000`
BG Tile Map: use 9800 Region

20.44.2.86 LCDCF_BG9C00 `#define LCDCF_BG9C00 0b00001000`
BG Tile Map: use 9C00 Region

20.44.2.87 LCDCF_OBJ8 `#define LCDCF_OBJ8 0b00000000`
Sprites Size: 8x8 pixels

20.44.2.88 LCDCF_OBJ16 `#define LCDCF_OBJ16 0b00000100`
Sprites Size: 8x16 pixels

20.44.2.89 LCDCF_OBJOFF `#define LCDCF_OBJOFF 0b00000000`
Sprites Display: Hidden

20.44.2.90 LCDCF_OBJON `#define LCDCF_OBJON 0b00000010`
Sprites Display: Visible

20.44.2.91 LCDCF_BG OFF `#define LCDCF_BG OFF 0b00000000`
Background Display: Hidden

20.44.2.92 LCDCF_BG ON `#define LCDCF_BG ON 0b00000001`
Background Display: Visible

20.44.2.93 LCDCF_B_ON `#define LCDCF_B_ON 7`
Bit for LCD On/Off Select

20.44.2.94 LCDCF_B_WIN9C00 `#define LCDCF_B_WIN9C00 6`
Bit for Window Tile Map Region Select

20.44.2.95 LCDCF_B_WINON `#define LCDCF_B_WINON 5`
Bit for Window Display On/Off Control

20.44.2.96 LCDCF_B_BG8000 `#define LCDCF_B_BG8000 4`
Bit for BG & Window Tile Data Region Select

20.44.2.97 LCDCF_B_BG9C00 `#define LCDCF_B_BG9C00 3`
Bit for BG Tile Map Region Select

20.44.2.98 LCDCF_B_OBJ16 `#define LCDCF_B_OBJ16 2`
Bit for Sprites Size Select

20.44.2.99 LCDCF_B_OBJON `#define LCDCF_B_OBJON 1`
Bit for Sprites Display Visible/Hidden Select

20.44.2.100 LCDCF_B_BGON `#define LCDCF_B_BGON 0`
Bit for Background Display Visible/Hidden Select

20.44.2.101 rSTAT `#define rSTAT STAT_REG`

20.44.2.102 STATF_LYC `#define STATF_LYC 0b01000000`
STAT Interrupt: LYC=LY Coincidence Source Enable

20.44.2.103 STATF_MODE10 `#define STATF_MODE10 0b00100000`
STAT Interrupt: Mode 2 OAM Source Enable

20.44.2.104 STATF_MODE01 `#define STATF_MODE01 0b00010000`
STAT Interrupt: Mode 1 VBlank Source Enable

20.44.2.105 STATF_MODE00 `#define STATF_MODE00 0b00001000`
STAT Interrupt: Mode 0 HBlank Source Enable

20.44.2.106 STATF_LYCF `#define STATF_LYCF 0b00000100`
LYC=LY Coincidence Status Flag, Set when LY contains the same value as LYC

20.44.2.107 STATF_HBL `#define STATF_HBL 0b00000000`
Current LCD Mode is: 0, in H-Blank

20.44.2.108 STATF_VBL `#define STATF_VBL 0b00000001`

Current LCD Mode is: 1, in V-Blank

20.44.2.109 STATF_OAM `#define STATF_OAM 0b00000010`

Current LCD Mode is: 2, in OAM-RAM is used by system (Searching OAM)

20.44.2.110 STATF_LCD `#define STATF_LCD 0b00000011`

Current LCD Mode is: 3, both OAM and VRAM used by system (Transferring Data to LCD Controller)

20.44.2.111 STATF_BUSY `#define STATF_BUSY 0b00000010`

When set, VRAM access is unsafe

20.44.2.112 STATF_B_LYC `#define STATF_B_LYC 6`

Bit for STAT Interrupt: LYC=LY Coincidence Source Enable

20.44.2.113 STATF_B_MODE10 `#define STATF_B_MODE10 5`

Bit for STAT Interrupt: Mode 2 OAM Source Enable

20.44.2.114 STATF_B_MODE01 `#define STATF_B_MODE01 4`

Bit for STAT Interrupt: Mode 1 VBlank Source Enable

20.44.2.115 STATF_B_MODE00 `#define STATF_B_MODE00 3`

Bit for STAT Interrupt: Mode 0 HBlank Source Enable

20.44.2.116 STATF_B_LYCF `#define STATF_B_LYCF 2`

Bit for LYC=LY Coincidence Status Flag

20.44.2.117 STATF_B_VBL `#define STATF_B_VBL 0`

20.44.2.118 STATF_B_OAM `#define STATF_B_OAM 1`

20.44.2.119 STATF_B_BUSY `#define STATF_B_BUSY 1`

Bit for when VRAM access is unsafe

20.44.2.120 rSCY `#define rSCY`

20.44.2.121 rSCX `#define rSCX SCX_REG`

20.44.2.122 rLY `#define rLY LY_REG`

20.44.2.123 rLYC `#define rLYC LYC_REG`

20.44.2.124 rDMA `#define rDMA DMA_REG`

20.44.2.125 rBGP `#define rBGP BGP_REG`

20.44.2.126 rOBP0 `#define rOBP0 OBP0_REG`

20.44.2.127 rOBP1 `#define rOBP1 OBP1_REG`

20.44.2.128 rWY `#define rWY WY_REG`

20.44.2.129 rWX `#define rWX WX_REG`

20.44.2.130 rKEY1 `#define rKEY1 KEY1_REG`

20.44.2.131 rSPD `#define rSPD KEY1_REG`

20.44.2.132 KEY1F_DBLSPD `#define KEY1F_DBLSPD 0b10000000`

20.44.2.133 KEY1F_PREPARE `#define KEY1F_PREPARE 0b00000001`

20.44.2.134 rVBK `#define rVBK VBK_REG`

20.44.2.135 VBK_BANK_0 `#define VBK_BANK_0 0`
Select Regular Map and Normal Tiles (CGB Mode Only)

20.44.2.136 VBK_TILES `#define VBK_TILES 0`
Select Regular Map and Normal Tiles (CGB Mode Only)

20.44.2.137 VBK_BANK_1 `#define VBK_BANK_1 1`
Select Map Attributes and Extra Tile Bank (CGB Mode Only)

20.44.2.138 VBK_ATTRIBUTES `#define VBK_ATTRIBUTES 1`
Select Map Attributes and Extra Tile Bank (CGB Mode Only)

20.44.2.139 BKGF_PRI `#define BKGF_PRI 0b10000000`
Background CGB BG and Window over Sprite priority Enabled

20.44.2.140 BKGF_YFLIP `#define BKGF_YFLIP 0b01000000`
Background CGB Y axis flip: Vertically mirrored

20.44.2.141 BKGF_XFLIP `#define BKGF_XFLIP 0b00100000`
Background CGB X axis flip: Horizontally mirrored

20.44.2.142 BKGF_BANK0 `#define BKGF_BANK0 0b00000000`
Background CGB Tile VRAM-Bank: Use Bank 0 (CGB Mode Only)

20.44.2.143 BKGF_BANK1 `#define BKGF_BANK1 0b00001000`
Background CGB Tile VRAM-Bank: Use Bank 1 (CGB Mode Only)

20.44.2.144 BKGF_CGB_PAL0 `#define BKGF_CGB_PAL0 0b00000000`
Background CGB Palette number (CGB Mode Only)

20.44.2.145 BKGF_CGB_PAL1 `#define BKGF_CGB_PAL1 0b00000001`
Background CGB Palette number (CGB Mode Only)

20.44.2.146 BKGF_CGB_PAL2 `#define BKGF_CGB_PAL2 0b00000010`
Background CGB Palette number (CGB Mode Only)

20.44.2.147 BKGF_CGB_PAL3 `#define BKGF_CGB_PAL3 0b00000011`
Background CGB Palette number (CGB Mode Only)

20.44.2.148 BKGF_CGB_PAL4 `#define BKGF_CGB_PAL4 0b00000100`
Background CGB Palette number (CGB Mode Only)

20.44.2.149 BKGF_CGB_PAL5 `#define BKGF_CGB_PAL5 0b00000101`
Background CGB Palette number (CGB Mode Only)

20.44.2.150 BKGF_CGB_PAL6 `#define BKGF_CGB_PAL6 0b00000110`
Background CGB Palette number (CGB Mode Only)

20.44.2.151 BKGF_CGB_PAL7 `#define BKGF_CGB_PAL7 0b00000111`
Background CGB Palette number (CGB Mode Only)

20.44.2.152 rHDMA1 `#define rHDMA1 HDMA1_REG`

20.44.2.153 rHDMA2 `#define rHDMA2 HDMA2_REG`

20.44.2.154 rHDMA3 `#define rHDMA3 HDMA3_REG`

20.44.2.155 rHDMA4 `#define rHDMA4 HDMA4_REG`

20.44.2.156 rHDMA5 `#define rHDMA5 HDMA5_REG`

20.44.2.157 HDMA5F_MODE_GP `#define HDMA5F_MODE_GP 0b00000000`

20.44.2.158 HDMA5F_MODE_HBL `#define HDMA5F_MODE_HBL 0b10000000`

20.44.2.159 HDMA5F_BUSY `#define HDMA5F_BUSY 0b10000000`

20.44.2.160 rRP `#define rRP RP_REG`

20.44.2.161 RPF_ENREAD `#define RPF_ENREAD 0b11000000`

20.44.2.162 RPF_DATAIN #define RPF_DATAIN 0b00000010

20.44.2.163 RPF_WRITE_HI #define RPF_WRITE_HI 0b00000001

20.44.2.164 RPF_WRITE_LO #define RPF_WRITE_LO 0b00000000

20.44.2.165 rBCPS #define rBCPS [BCPS_REG](#)

20.44.2.166 BCPSF_AUTOINC #define BCPSF_AUTOINC 0b10000000

20.44.2.167 rBCPD #define rBCPD [BCPD_REG](#)

20.44.2.168 rOCPS #define rOCPS [OCPS_REG](#)

20.44.2.169 OCPSF_AUTOINC #define OCPSF_AUTOINC 0b10000000

20.44.2.170 rOCPD #define rOCPD [OCPD_REG](#)

20.44.2.171 rSVBK #define rSVBK [SVBK_REG](#)

20.44.2.172 rSMBK #define rSMBK [SVBK_REG](#)

20.44.2.173 rPCM12 #define rPCM12 [PCM12_REG](#)

20.44.2.174 rPCM34 #define rPCM34 [PCM34_REG](#)

20.44.2.175 rIE #define rIE [IE_REG](#)

20.44.2.176 IEF_HILO #define IEF_HILO 0b00010000

20.44.2.177 IEF_SERIAL #define IEF_SERIAL 0b00001000

20.44.2.178 IEF_TIMER #define IEF_TIMER 0b00000100

20.44.2.179 IEF_STAT #define IEF_STAT 0b00000010

20.44.2.180 IEF_VBLANK `#define IEF_VBLANK 0b00000001`

20.44.2.181 AUDLEN_DUTY_12_5 `#define AUDLEN_DUTY_12_5 0b00000000`

20.44.2.182 AUDLEN_DUTY_25 `#define AUDLEN_DUTY_25 0b01000000`

20.44.2.183 AUDLEN_DUTY_50 `#define AUDLEN_DUTY_50 0b10000000`

20.44.2.184 AUDLEN_DUTY_75 `#define AUDLEN_DUTY_75 0b11000000`

20.44.2.185 AUDLEN_LENGTH `#define AUDLEN_LENGTH(
x) (x)`

20.44.2.186 AUDENV_VOL `#define AUDENV_VOL(
x) ((x) << 4)`

20.44.2.187 AUDENV_UP `#define AUDENV_UP 0b00001000`

20.44.2.188 AUDENV_DOWN `#define AUDENV_DOWN 0b00000000`

20.44.2.189 AUDENV_LENGTH `#define AUDENV_LENGTH(
x) (x)`

20.44.2.190 AUDHIGH_RESTART `#define AUDHIGH_RESTART 0b10000000`

20.44.2.191 AUDHIGH_LENGTH_ON `#define AUDHIGH_LENGTH_ON 0b01000000`

20.44.2.192 AUDHIGH_LENGTH_OFF `#define AUDHIGH_LENGTH_OFF 0b00000000`

20.44.2.193 OAMF_PRI `#define OAMF_PRI 0b10000000`
BG and Window over Sprite Enabled

20.44.2.194 OAMF_YFLIP `#define OAMF_YFLIP 0b01000000`
Sprite Y axis flip: Vertically mirrored

20.44.2.195 OAMF_XFLIP `#define OAMF_XFLIP 0b00100000`
Sprite X axis flip: Horizontally mirrored

20.44.2.196 OAMF_PAL0 `#define OAMF_PAL0 0b00000000`
Sprite Palette number: use OBP0 (Non-CGB Mode Only)

20.44.2.197 OAMF_PAL1 `#define OAMF_PAL1 0b00010000`
Sprite Palette number: use OBP1 (Non-CGB Mode Only)

20.44.2.198 OAMF_BANK0 `#define OAMF_BANK0 0b00000000`
Sprite Tile VRAM-Bank: Use Bank 0 (CGB Mode Only)

20.44.2.199 OAMF_BANK1 `#define OAMF_BANK1 0b00001000`
Sprite Tile VRAM-Bank: Use Bank 1 (CGB Mode Only)

20.44.2.200 OAMF_CGB_PAL0 `#define OAMF_CGB_PAL0 0b00000000`
Sprite CGB Palette number: use OCP0 (CGB Mode Only)

20.44.2.201 OAMF_CGB_PAL1 `#define OAMF_CGB_PAL1 0b00000001`
Sprite CGB Palette number: use OCP1 (CGB Mode Only)

20.44.2.202 OAMF_CGB_PAL2 `#define OAMF_CGB_PAL2 0b00000010`
Sprite CGB Palette number: use OCP2 (CGB Mode Only)

20.44.2.203 OAMF_CGB_PAL3 `#define OAMF_CGB_PAL3 0b00000011`
Sprite CGB Palette number: use OCP3 (CGB Mode Only)

20.44.2.204 OAMF_CGB_PAL4 `#define OAMF_CGB_PAL4 0b00000100`
Sprite CGB Palette number: use OCP4 (CGB Mode Only)

20.44.2.205 OAMF_CGB_PAL5 `#define OAMF_CGB_PAL5 0b00000101`
Sprite CGB Palette number: use OCP5 (CGB Mode Only)

20.44.2.206 OAMF_CGB_PAL6 `#define OAMF_CGB_PAL6 0b00000110`
Sprite CGB Palette number: use OCP6 (CGB Mode Only)

20.44.2.207 OAMF_CGB_PAL7 `#define OAMF_CGB_PAL7 0b00000111`
Sprite CGB Palette number: use OCP7 (CGB Mode Only)

20.44.2.208 OAMF_PALMASK `#define OAMF_PALMASK 0b00000111`
Mask for Sprite CGB Palette number (CGB Mode Only)

20.44.2.209 DEVICE_SCREEN_X_OFFSET `#define DEVICE_SCREEN_X_OFFSET 0`
Offset of visible screen (in tile units) from left edge of hardware map

20.44.2.210 DEVICE_SCREEN_Y_OFFSET `#define DEVICE_SCREEN_Y_OFFSET 0`
Offset of visible screen (in tile units) from top edge of hardware map

20.44.2.211 DEVICE_SCREEN_WIDTH `#define DEVICE_SCREEN_WIDTH 20`
Width of visible screen in tile units

20.44.2.212 DEVICE_SCREEN_HEIGHT `#define DEVICE_SCREEN_HEIGHT 18`
Height of visible screen in tile units

20.44.2.213 DEVICE_SCREEN_BUFFER_WIDTH `#define DEVICE_SCREEN_BUFFER_WIDTH 32`
Width of hardware map buffer in tile units

20.44.2.214 DEVICE_SCREEN_BUFFER_HEIGHT `#define DEVICE_SCREEN_BUFFER_HEIGHT 32`
Height of hardware map buffer in tile units

20.44.2.215 **DEVICE_SCREEN_MAP_ENTRY_SIZE** `#define DEVICE_SCREEN_MAP_ENTRY_SIZE 1`
 Number of bytes per hardware map entry

20.44.2.216 **DEVICE_SPRITE_PX_OFFSET_X** `#define DEVICE_SPRITE_PX_OFFSET_X 8`
 Offset of sprite X coordinate origin (in pixels) from left edge of visible screen

20.44.2.217 **DEVICE_SPRITE_PX_OFFSET_Y** `#define DEVICE_SPRITE_PX_OFFSET_Y 16`
 Offset of sprite Y coordinate origin (in pixels) from top edge of visible screen

20.44.2.218 **DEVICE_WINDOW_PX_OFFSET_X** `#define DEVICE_WINDOW_PX_OFFSET_X 7`
 Minimal X coordinate of the window layer

20.44.2.219 **DEVICE_WINDOW_PX_OFFSET_Y** `#define DEVICE_WINDOW_PX_OFFSET_Y 0`
 Minimal Y coordinate of the window layer

20.44.2.220 **DEVICE_SCREEN_PX_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`
 Width of visible screen in pixels

20.44.2.221 **DEVICE_SCREEN_PX_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`
 Height of visible screen in pixels

20.44.3 Variable Documentation

20.44.3.1 **_VRAM** `__BYTES _VRAM[]`
 Memory map

20.44.3.2 **_VRAM8000** `__BYTES _VRAM8000[]`

20.44.3.3 **_VRAM8800** `__BYTES _VRAM8800[]`

20.44.3.4 **_VRAM9000** `__BYTES _VRAM9000[]`

20.44.3.5 **_SCRN0** `__BYTES _SCRN0[]`

20.44.3.6 **_SCRN1** `__BYTES _SCRN1[]`

20.44.3.7 **_SRAM** `__BYTES _SRAM[]`

20.44.3.8 **_RAM** `__BYTES _RAM[]`

20.44.3.9 **_RAMBANK** `__BYTES _RAMBANK[]`

20.44.3.10 **_OAMRAM** [__BYTES](#) _OAMRAM[]

20.44.3.11 **_IO** [__BYTE_REG](#) _IO[]

20.44.3.12 **_AUD3WAVERAM** [__BYTE_REG](#) _AUD3WAVERAM[]

20.44.3.13 **_HRAM** [__BYTE_REG](#) _HRAM[]

20.44.3.14 **rRAMG** [__BYTE_REG](#) rRAMG
MBC5 registers

20.44.3.15 **rROMB0** [__BYTE_REG](#) rROMB0

20.44.3.16 **rROMB1** [__BYTE_REG](#) rROMB1

20.44.3.17 **rRAMB** [__BYTE_REG](#) rRAMB

20.44.3.18 **P1_REG** [__REG](#) P1_REG
IO Registers Joystick: 1.1.P15.P14.P13.P12.P11.P10

20.44.3.19 **SB_REG** [__REG](#) SB_REG
Serial IO data buffer

20.44.3.20 **SC_REG** [__REG](#) SC_REG
Serial IO control register

20.44.3.21 **DIV_REG** [__REG](#) DIV_REG
Divider register

20.44.3.22 **TIMA_REG** [__REG](#) TIMA_REG
Timer counter

20.44.3.23 **TMA_REG** [__REG](#) TMA_REG
Timer modulo

20.44.3.24 **TAC_REG** [__REG](#) TAC_REG
Timer control

20.44.3.25 **IF_REG** [__REG](#) IF_REG
Interrupt flags: 0.0.0.JOY.SIO.TIM.LCD.VBL

20.44.3.26 **NR10_REG** [__REG](#) NR10_REG
Sound Channel 1 Sweep

20.44.3.27 **NR11_REG** [__REG](#) NR11_REG
Sound Channel 1 Sound length/Wave pattern duty

20.44.3.28 NR12_REG [__REG](#) NR12_REG
Sound Channel 1 Volume Envelope

20.44.3.29 NR13_REG [__REG](#) NR13_REG
Sound Channel 1 Frequency Low

20.44.3.30 NR14_REG [__REG](#) NR14_REG
Sound Channel 1 Frequency High

20.44.3.31 NR21_REG [__REG](#) NR21_REG
Sound Channel 2 Tone

20.44.3.32 NR22_REG [__REG](#) NR22_REG
Sound Channel 2 Volume Envelope

20.44.3.33 NR23_REG [__REG](#) NR23_REG
Sound Channel 2 Frequency data Low

20.44.3.34 NR24_REG [__REG](#) NR24_REG
Sound Channel 2 Frequency data High

20.44.3.35 NR30_REG [__REG](#) NR30_REG
Sound Channel 3 Sound on/off

20.44.3.36 NR31_REG [__REG](#) NR31_REG
Sound Channel 3 Sound Length

20.44.3.37 NR32_REG [__REG](#) NR32_REG
Sound Channel 3 Select output level

20.44.3.38 NR33_REG [__REG](#) NR33_REG
Sound Channel 3 Frequency data Low

20.44.3.39 NR34_REG [__REG](#) NR34_REG
Sound Channel 3 Frequency data High

20.44.3.40 NR41_REG [__REG](#) NR41_REG
Sound Channel 4 Sound Length

20.44.3.41 NR42_REG [__REG](#) NR42_REG
Sound Channel 4 Volume Envelope

20.44.3.42 NR43_REG [__REG](#) NR43_REG
Sound Channel 4 Polynomial Counter

20.44.3.43 NR44_REG [__REG](#) NR44_REG
Sound Channel 4 Counter / Consecutive and Initial

20.44.3.44 NR50_REG [__REG](#) NR50_REG
Sound Channel control / ON-OFF / Volume

20.44.3.45 NR51_REG [__REG](#) NR51_REG
Sound Selection of Sound output terminal

20.44.3.46 NR52_REG [__REG](#) NR52_REG
Sound Master on/off

20.44.3.47 AUD3WAVE [__BYTE_REG](#) AUD3WAVE[16]

20.44.3.48 PCM_SAMPLE [__BYTE_REG](#) PCM_SAMPLE[16]

20.44.3.49 LCDC_REG [__REG](#) LCDC_REG
LCD control

20.44.3.50 STAT_REG [__REG](#) STAT_REG
LCD status

20.44.3.51 SCY_REG [__REG](#) SCY_REG
Scroll Y

20.44.3.52 SCX_REG [__REG](#) SCX_REG
Scroll X

20.44.3.53 LY_REG [__REG](#) LY_REG
LCDC Y-coordinate

20.44.3.54 LYC_REG [__REG](#) LYC_REG
LY compare

20.44.3.55 DMA_REG [__REG](#) DMA_REG
DMA transfer

20.44.3.56 BGP_REG [__REG](#) BGP_REG
BG palette data

20.44.3.57 OBP0_REG [__REG](#) OBP0_REG
OBJ palette 0 data

20.44.3.58 OBP1_REG [__REG](#) OBP1_REG
OBJ palette 1 data

20.44.3.59 WY_REG [__REG](#) WY_REG
Window Y coordinate

20.44.3.60 WX_REG [__REG](#) WX_REG
Window X coordinate

20.44.3.61 KEY1_REG [__REG](#) KEY1_REG
CPU speed

20.44.3.62 VBK_REG [__REG](#) VBK_REG
VRAM bank select (CGB only)

See also

[VBK_BANK_0](#), [VBK_TILES](#), [VBK_BANK_1](#), [VBK_ATTRIBUTES](#)

20.44.3.63 HDMA1_REG [__REG](#) HDMA1_REG
DMA control 1

20.44.3.64 HDMA2_REG [__REG](#) HDMA2_REG
DMA control 2

20.44.3.65 HDMA3_REG [__REG](#) HDMA3_REG
DMA control 3

20.44.3.66 HDMA4_REG [__REG](#) HDMA4_REG
DMA control 4

20.44.3.67 HDMA5_REG [__REG](#) HDMA5_REG
DMA control 5

20.44.3.68 RP_REG [__REG](#) RP_REG
IR port

20.44.3.69 BCPS_REG [__REG](#) BCPS_REG
BG color palette specification

20.44.3.70 BCPD_REG [__REG](#) BCPD_REG
BG color palette data

20.44.3.71 OCPS_REG [__REG](#) OCPS_REG
OBJ color palette specification

20.44.3.72 OCPD_REG [__REG](#) OCPD_REG
OBJ color palette data

20.44.3.73 SVBK_REG [__REG](#) SVBK_REG
WRAM bank

20.44.3.74 PCM12_REG [__REG](#) PCM12_REG
Sound channel 1&2 PCM amplitude (R)

20.44.3.75 PCM34_REG [__REG](#) PCM34_REG
Sound channel 3&4 PCM amplitude (R)

20.44.3.76 IE_REG [__REG](#) IE_REG
Interrupt enable

20.45 gbdk-lib/include/msx/hardware.h File Reference

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

Macros

- `#define` [__BYTES](#) extern [UBYTE](#)
- `#define` [__BYTE_REG](#) extern volatile [UBYTE](#)
- `#define` [MEMCTL_JOYON](#) 0b00000000
- `#define` [MEMCTL_JOYOFF](#) 0b00000100
- `#define` [MEMCTL_BASEON](#) 0b00000000
- `#define` [MEMCTL_BASEOFF](#) 0b00001000

- #define MEMCTL_RAMON 0b00000000
- #define MEMCTL_RAMOFF 0b00010000
- #define MEMCTL_CROMON 0b00000000
- #define MEMCTL_CROMOFF 0b00100000
- #define MEMCTL_ROMON 0b00000000
- #define MEMCTL_ROMOFF 0b01000000
- #define MEMCTL_EXTON 0b00000000
- #define MEMCTL_EXTOFF 0b10000000
- #define JOY_P1_LATCH 0b00000010
- #define JOY_P2_LATCH 0b00001000
- #define PSG_LATCH 0x80
- #define PSG_CH0 0b00000000
- #define PSG_CH1 0b00100000
- #define PSG_CH2 0b01000000
- #define PSG_CH3 0b01100000
- #define PSG_VOLUME 0b00010000
- #define STATF_INT_VBL 0b10000000
- #define STATF_9_SPR 0b01000000
- #define STATF_SPR_COLL 0b00100000
- #define VDP_REG_MASK 0b10000000
- #define VDP_R0 0b10000000
- #define R0_DEFAULT 0b00000000
- #define R0_CB_OUTPUT 0b00000000
- #define R0_CB_INPUT 0b01000000
- #define R0_IE2_OFF 0b00000000
- #define R0_IE2 0b00100000
- #define R0_IE1_OFF 0b00000000
- #define R0_IE1 0b00010000
- #define R0_SCR_MODE1 0b00000000
- #define R0_SCR_MODE2 0b000000010
- #define R0_SCR_MODE3 0b000000100
- #define R0_ES_OFF 0b00000000
- #define R0_ES 0b00000001
- #define VDP_R1 0b10000001
- #define R1_DEFAULT 0b10000000
- #define R1_DISP_OFF 0b00000000
- #define R1_DISP_ON 0b01000000
- #define R1_IE_OFF 0b00000000
- #define R1_IE 0b00100000
- #define R1_SCR_MODE1 0b00010000
- #define R1_SCR_MODE2 0b00000000
- #define R1_SCR_MODE3 0b00000000
- #define R1_SPR_8X8 0b00000000
- #define R1_SPR_16X16 0b000000010
- #define R1_SPR_MAG 0b00000001
- #define R1_SPR_MAG_OFF 0b00000000
- #define VDP_R2 0b10000010
- #define R2_MAP_0x3800 0xFF
- #define R2_MAP_0x3000 0xFD
- #define R2_MAP_0x2800 0xFB
- #define R2_MAP_0x2000 0xF9
- #define R2_MAP_0x1800 0xF7
- #define R2_MAP_0x1000 0xF5
- #define R2_MAP_0x0800 0xF3
- #define R2_MAP_0x0000 0xF1

- #define [VDP_R3](#) 0b10000011
- #define [VDP_R4](#) 0b10000100
- #define [VDP_R5](#) 0b10000101
- #define [R5_SAT_0x3F00](#) 0xFF
- #define [R5_SAT_MASK](#) 0b10000001
- #define [VDP_R6](#) 0b10000110
- #define [R6_BANK0](#) 0xFB
- #define [R6_DATA_0x0000](#) 0xFB
- #define [R6_BANK1](#) 0xFF
- #define [R6_DATA_0x2000](#) 0xFF
- #define [VDP_R7](#) 0b10000111
- #define [VDP_RBORDER](#) 0b10000111
- #define [R7_COLOR_MASK](#) 0b11110000
- #define [VDP_R8](#) 0b10001000
- #define [VDP_RSCX](#) 0b10001000
- #define [VDP_R9](#) 0b10001001
- #define [VDP_RSCY](#) 0b10001001
- #define [VDP_R10](#) 0b10001010
- #define [R10_INT_OFF](#) 0xFF
- #define [R10_INT_EVERY](#) 0x00
- #define [JOY_P1_UP](#) 0b00000001
- #define [JOY_P1_DOWN](#) 0b00000010
- #define [JOY_P1_LEFT](#) 0b00000100
- #define [JOY_P1_RIGHT](#) 0b00001000
- #define [JOY_P1_SW1](#) 0b00010000
- #define [JOY_P1_TRIGGER](#) 0b00010000
- #define [JOY_P1_SW2](#) 0b00100000
- #define [JOY_P2_UP](#) 0b01000000
- #define [JOY_P2_DOWN](#) 0b10000000
- #define [JOY_P2_LEFT](#) 0b00000001
- #define [JOY_P2_RIGHT](#) 0b00000010
- #define [JOY_P2_SW1](#) 0b00000100
- #define [JOY_P2_TRIGGER](#) 0b00000100
- #define [JOY_P2_SW2](#) 0b00001000
- #define [JOY_RESET](#) 0b00010000
- #define [JOY_P1_LIGHT](#) 0b01000000
- #define [JOY_P2_LIGHT](#) 0b10000000
- #define [SYSTEM_PAL](#) 0x00
- #define [SYSTEM_NTSC](#) 0x01
- #define [VBK_TILES](#) 0
- #define [VBK_ATTRIBUTES](#) 1
- #define [VDP_SAT_TERM](#) 0xD0
- #define [DEVICE_SCREEN_PX_WIDTH](#) ([DEVICE_SCREEN_WIDTH](#) * 8)
- #define [DEVICE_SCREEN_PX_HEIGHT](#) ([DEVICE_SCREEN_HEIGHT](#) * 8)

Variables

- [UBYTE shadow_VDP_R0](#)
- [UBYTE shadow_VDP_R1](#)
- [UBYTE shadow_VDP_R2](#)
- [UBYTE shadow_VDP_R3](#)
- [UBYTE shadow_VDP_R4](#)
- [UBYTE shadow_VDP_R5](#)
- [UBYTE shadow_VDP_R6](#)
- [UBYTE shadow_VDP_R7](#)

- UBYTE shadow_VDP_RBORDER
- UBYTE shadow_VDP_R8
- UBYTE shadow_VDP_RSCX
- UBYTE shadow_VDP_R9
- UBYTE shadow_VDP_RSCY
- UBYTE shadow_VDP_R10
- const UBYTE _BIOS
- const UBYTE _SYSTEM
- volatile UBYTE VDP_ATTR_SHIFT

20.45.1 Detailed Description

Defines that let the MSX hardware registers be accessed from C.

20.45.2 Macro Definition Documentation

20.45.2.1 __BYTES `#define __BYTES extern UBYTE`

20.45.2.2 __BYTE_REG `#define __BYTE_REG extern volatile UBYTE`

20.45.2.3 MEMCTL_JOYON `#define MEMCTL_JOYON 0b00000000`

20.45.2.4 MEMCTL_JOYOFF `#define MEMCTL_JOYOFF 0b00000100`

20.45.2.5 MEMCTL_BASEON `#define MEMCTL_BASEON 0b00000000`

20.45.2.6 MEMCTL_BASEOFF `#define MEMCTL_BASEOFF 0b00001000`

20.45.2.7 MEMCTL_RAMON `#define MEMCTL_RAMON 0b00000000`

20.45.2.8 MEMCTL_RAMOFF `#define MEMCTL_RAMOFF 0b00010000`

20.45.2.9 MEMCTL_CROMON `#define MEMCTL_CROMON 0b00000000`

20.45.2.10 MEMCTL_CROMOFF `#define MEMCTL_CROMOFF 0b00100000`

20.45.2.11 MEMCTL_ROMON `#define MEMCTL_ROMON 0b00000000`

20.45.2.12 MEMCTL_ROMOFF `#define MEMCTL_ROMOFF 0b01000000`

- 20.45.2.13 MEMCTL_EXTON** `#define MEMCTL_EXTON 0b00000000`
- 20.45.2.14 MEMCTL_EXTOFF** `#define MEMCTL_EXTOFF 0b10000000`
- 20.45.2.15 JOY_P1_LATCH** `#define JOY_P1_LATCH 0b00000010`
- 20.45.2.16 JOY_P2_LATCH** `#define JOY_P2_LATCH 0b00001000`
- 20.45.2.17 PSG_LATCH** `#define PSG_LATCH 0x80`
- 20.45.2.18 PSG_CH0** `#define PSG_CH0 0b00000000`
- 20.45.2.19 PSG_CH1** `#define PSG_CH1 0b00100000`
- 20.45.2.20 PSG_CH2** `#define PSG_CH2 0b01000000`
- 20.45.2.21 PSG_CH3** `#define PSG_CH3 0b01100000`
- 20.45.2.22 PSG_VOLUME** `#define PSG_VOLUME 0b00010000`
- 20.45.2.23 STATF_INT_VBL** `#define STATF_INT_VBL 0b10000000`
- 20.45.2.24 STATF_9_SPR** `#define STATF_9_SPR 0b01000000`
- 20.45.2.25 STATF_SPR_COLL** `#define STATF_SPR_COLL 0b00100000`
- 20.45.2.26 VDP_REG_MASK** `#define VDP_REG_MASK 0b10000000`
- 20.45.2.27 VDP_R0** `#define VDP_R0 0b10000000`
- 20.45.2.28 R0_DEFAULT** `#define R0_DEFAULT 0b00000000`
- 20.45.2.29 R0_CB_OUTPUT** `#define R0_CB_OUTPUT 0b00000000`
- 20.45.2.30 R0_CB_INPUT** `#define R0_CB_INPUT 0b01000000`

20.45.2.31 R0_IE2_OFF #define R0_IE2_OFF 0b00000000

20.45.2.32 R0_IE2 #define R0_IE2 0b00100000

20.45.2.33 R0_IE1_OFF #define R0_IE1_OFF 0b00000000

20.45.2.34 R0_IE1 #define R0_IE1 0b00010000

20.45.2.35 R0_SCR_MODE1 #define R0_SCR_MODE1 0b00000000

20.45.2.36 R0_SCR_MODE2 #define R0_SCR_MODE2 0b00000010

20.45.2.37 R0_SCR_MODE3 #define R0_SCR_MODE3 0b00000100

20.45.2.38 R0_ES_OFF #define R0_ES_OFF 0b00000000

20.45.2.39 R0_ES #define R0_ES 0b00000001

20.45.2.40 VDP_R1 #define VDP_R1 0b10000001

20.45.2.41 R1_DEFAULT #define R1_DEFAULT 0b10000000

20.45.2.42 R1_DISP_OFF #define R1_DISP_OFF 0b00000000

20.45.2.43 R1_DISP_ON #define R1_DISP_ON 0b01000000

20.45.2.44 R1_IE_OFF #define R1_IE_OFF 0b00000000

20.45.2.45 R1_IE #define R1_IE 0b00100000

20.45.2.46 R1_SCR_MODE1 #define R1_SCR_MODE1 0b00010000

20.45.2.47 R1_SCR_MODE2 #define R1_SCR_MODE2 0b00000000

20.45.2.48 R1_SCR_MODE3 #define R1_SCR_MODE3 0b00000000

20.45.2.49 R1_SPR_8X8 `#define R1_SPR_8X8 0b00000000`

20.45.2.50 R1_SPR_16X16 `#define R1_SPR_16X16 0b00000010`

20.45.2.51 R1_SPR_MAG `#define R1_SPR_MAG 0b00000001`

20.45.2.52 R1_SPR_MAG_OFF `#define R1_SPR_MAG_OFF 0b00000000`

20.45.2.53 VDP_R2 `#define VDP_R2 0b10000010`

20.45.2.54 R2_MAP_0x3800 `#define R2_MAP_0x3800 0xFF`

20.45.2.55 R2_MAP_0x3000 `#define R2_MAP_0x3000 0xFD`

20.45.2.56 R2_MAP_0x2800 `#define R2_MAP_0x2800 0xFB`

20.45.2.57 R2_MAP_0x2000 `#define R2_MAP_0x2000 0xF9`

20.45.2.58 R2_MAP_0x1800 `#define R2_MAP_0x1800 0xF7`

20.45.2.59 R2_MAP_0x1000 `#define R2_MAP_0x1000 0xF5`

20.45.2.60 R2_MAP_0x0800 `#define R2_MAP_0x0800 0xF3`

20.45.2.61 R2_MAP_0x0000 `#define R2_MAP_0x0000 0xF1`

20.45.2.62 VDP_R3 `#define VDP_R3 0b10000011`

20.45.2.63 VDP_R4 `#define VDP_R4 0b10000100`

20.45.2.64 VDP_R5 `#define VDP_R5 0b10000101`

20.45.2.65 R5_SAT_0x3F00 `#define R5_SAT_0x3F00 0xFF`

20.45.2.66 R5_SAT_MASK `#define R5_SAT_MASK 0b10000001`

20.45.2.67 VDP_R6 `#define VDP_R6 0b10000110`

20.45.2.68 R6_BANK0 `#define R6_BANK0 0xFB`

20.45.2.69 R6_DATA_0x0000 `#define R6_DATA_0x0000 0xFB`

20.45.2.70 R6_BANK1 `#define R6_BANK1 0xFF`

20.45.2.71 R6_DATA_0x2000 `#define R6_DATA_0x2000 0xFF`

20.45.2.72 VDP_R7 `#define VDP_R7 0b10000111`

20.45.2.73 VDP_RBORDER `#define VDP_RBORDER 0b10000111`

20.45.2.74 R7_COLOR_MASK `#define R7_COLOR_MASK 0b11110000`

20.45.2.75 VDP_R8 `#define VDP_R8 0b10001000`

20.45.2.76 VDP_RSCX `#define VDP_RSCX 0b10001000`

20.45.2.77 VDP_R9 `#define VDP_R9 0b10001001`

20.45.2.78 VDP_RSCY `#define VDP_RSCY 0b10001001`

20.45.2.79 VDP_R10 `#define VDP_R10 0b10001010`

20.45.2.80 R10_INT_OFF `#define R10_INT_OFF 0xFF`

20.45.2.81 R10_INT EVERY `#define R10_INT EVERY 0x00`

20.45.2.82 JOY_P1_UP `#define JOY_P1_UP 0b00000001`

20.45.2.83 JOY_P1_DOWN `#define JOY_P1_DOWN 0b00000010`

20.45.2.84 JOY_P1_LEFT `#define JOY_P1_LEFT 0b00000100`

20.45.2.85 JOY_P1_RIGHT `#define JOY_P1_RIGHT 0b00001000`

20.45.2.86 JOY_P1_SW1 `#define JOY_P1_SW1 0b00010000`

20.45.2.87 JOY_P1_TRIGGER `#define JOY_P1_TRIGGER 0b00010000`

20.45.2.88 JOY_P1_SW2 `#define JOY_P1_SW2 0b00100000`

20.45.2.89 JOY_P2_UP `#define JOY_P2_UP 0b01000000`

20.45.2.90 JOY_P2_DOWN `#define JOY_P2_DOWN 0b10000000`

20.45.2.91 JOY_P2_LEFT `#define JOY_P2_LEFT 0b00000001`

20.45.2.92 JOY_P2_RIGHT `#define JOY_P2_RIGHT 0b00000010`

20.45.2.93 JOY_P2_SW1 `#define JOY_P2_SW1 0b00000100`

20.45.2.94 JOY_P2_TRIGGER `#define JOY_P2_TRIGGER 0b00000100`

20.45.2.95 JOY_P2_SW2 `#define JOY_P2_SW2 0b00001000`

20.45.2.96 JOY_RESET `#define JOY_RESET 0b00010000`

20.45.2.97 JOY_P1_LIGHT `#define JOY_P1_LIGHT 0b01000000`

20.45.2.98 JOY_P2_LIGHT `#define JOY_P2_LIGHT 0b10000000`

20.45.2.99 SYSTEM_PAL `#define SYSTEM_PAL 0x00`

20.45.2.100 SYSTEM_NTSC `#define SYSTEM_NTSC 0x01`

20.45.2.101 VBK_TILES `#define VBK_TILES 0`

20.45.2.102 VBK_ATTRIBUTES `#define VBK_ATTRIBUTES 1`

20.45.2.103 VDP_SAT_TERM `#define VDP_SAT_TERM 0xD0`

20.45.2.104 DEVICE_SCREEN_PX_WIDTH `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

20.45.2.105 DEVICE_SCREEN_PX_HEIGHT `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

20.45.3 Variable Documentation

20.45.3.1 shadow_VDP_R0 `UBYTE shadow_VDP_R0`

20.45.3.2 shadow_VDP_R1 `UBYTE shadow_VDP_R1`

20.45.3.3 shadow_VDP_R2 `UBYTE shadow_VDP_R2`

20.45.3.4 shadow_VDP_R3 `UBYTE shadow_VDP_R3`

20.45.3.5 shadow_VDP_R4 `UBYTE shadow_VDP_R4`

20.45.3.6 shadow_VDP_R5 `UBYTE shadow_VDP_R5`

20.45.3.7 shadow_VDP_R6 `UBYTE shadow_VDP_R6`

20.45.3.8 shadow_VDP_R7 `UBYTE shadow_VDP_R7`

20.45.3.9 shadow_VDP_RBORDER `UBYTE shadow_VDP_RBORDER`

20.45.3.10 shadow_VDP_R8 `UBYTE shadow_VDP_R8`

20.45.3.11 shadow_VDP_RSCX `UBYTE shadow_VDP_RSCX`

20.45.3.12 shadow_VDP_R9 `UBYTE shadow_VDP_R9`

20.45.3.13 shadow_VDP_RSCY `UBYTE shadow_VDP_RSCY`

20.45.3.14 shadow_VDP_R10 `UBYTE shadow_VDP_R10`

20.45.3.15 `_BIOS` `const UBYTE _BIOS`

20.45.3.16 `_SYSTEM` `const UBYTE _SYSTEM`

20.45.3.17 `VDP_ATTR_SHIFT` `volatile UBYTE VDP_ATTR_SHIFT`

20.46 gbdk-lib/include/nes/hardware.h File Reference

`#include <types.h>`

Macros

- `#define __BYTES` extern `UBYTE`
- `#define __BYTE_REG` extern volatile `UBYTE`
- `#define PPUCTRL` `((uint8_t*)0x2000);`
- `#define PPUCTRL_NMI` `0b10000000`
- `#define PPUCTRL_SPR_8X8` `0b00000000`
- `#define PPUCTRL_SPR_8X16` `0b00100000`
- `#define PPUCTRL_BG_CHR` `0b00010000`
- `#define PPUCTRL_SPR_CHR` `0b00001000`
- `#define PPUCTRL_INC32` `0b00000100`
- `#define PPUMASK` `((uint8_t*)0x2001);`
- `#define PPUMASK_BLUE` `0b10000000`
- `#define PPUMASK_RED` `0b01000000`
- `#define PPUMASK_GREEN` `0b00100000`
- `#define PPUMASK_SHOW_SPR` `0b00010000`
- `#define PPUMASK_SHOW_BG` `0b00001000`
- `#define PPUMASK_SHOW_SPR_LC` `0b00000100`
- `#define PPUMASK_SHOW_BG_LC` `0b00000010`
- `#define PPUMASK_MONOCHROME` `0b00000001`
- `#define PPUSTATUS` `((uint8_t*)0x2002);`
- `#define OAMADDR` `((uint8_t*)0x2003);`
- `#define OAMDATA` `((uint8_t*)0x2004);`
- `#define PPU SCROLL` `((uint8_t*)0x2005);`
- `#define PPUADDR` `((uint8_t*)0x2006);`
- `#define PPUDATA` `((uint8_t*)0x2007);`
- `#define OAMDMA` `((uint8_t*)0x4014);`
- `#define DEVICE_SCREEN_X_OFFSET` `0`
- `#define DEVICE_SCREEN_Y_OFFSET` `0`
- `#define DEVICE_SCREEN_WIDTH` `32`
- `#define DEVICE_SCREEN_HEIGHT` `30`
- `#define DEVICE_SCREEN_BUFFER_WIDTH` `32`
- `#define DEVICE_SCREEN_BUFFER_HEIGHT` `30`
- `#define DEVICE_SCREEN_MAP_ENTRY_SIZE` `2`
- `#define DEVICE_SPRITE_PX_OFFSET_X` `0`
- `#define DEVICE_SPRITE_PX_OFFSET_Y` `-1`
- `#define DEVICE_WINDOW_PX_OFFSET_X` `0`
- `#define DEVICE_WINDOW_PX_OFFSET_Y` `0`
- `#define DEVICE_SCREEN_PX_WIDTH` `(DEVICE_SCREEN_WIDTH * 8)`
- `#define DEVICE_SCREEN_PX_HEIGHT` `(DEVICE_SCREEN_HEIGHT * 8)`

Variables

- volatile [UBYTE shadow_PPUCTRL](#)
- volatile [UBYTE shadow_PPUMASK](#)
- volatile [UBYTE bkg_scroll_x](#)
- volatile [UBYTE bkg_scroll_y](#)

20.46.1 Detailed Description

Defines that let the NES hardware registers be accessed from C.

20.46.2 Macro Definition Documentation

20.46.2.1 `__BYTES` `#define __BYTES extern UBYTE`

20.46.2.2 `__BYTE_REG` `#define __BYTE_REG extern volatile UBYTE`

20.46.2.3 `PPUCTRL` `#define PPUCTRL ((uint8_t*)0x2000);`

20.46.2.4 `PPUCTRL_NMI` `#define PPUCTRL_NMI 0b10000000`

20.46.2.5 `PPUCTRL_SPR_8X8` `#define PPUCTRL_SPR_8X8 0b00000000`

20.46.2.6 `PPUCTRL_SPR_8X16` `#define PPUCTRL_SPR_8X16 0b00100000`

20.46.2.7 `PPUCTRL_BG_CHR` `#define PPUCTRL_BG_CHR 0b00010000`

20.46.2.8 `PPUCTRL_SPR_CHR` `#define PPUCTRL_SPR_CHR 0b00001000`

20.46.2.9 `PPUCTRL_INC32` `#define PPUCTRL_INC32 0b00000100`

20.46.2.10 `PPUMASK` `#define PPUMASK ((uint8_t*)0x2001);`

20.46.2.11 `PPUMASK_BLUE` `#define PPUMASK_BLUE 0b10000000`

20.46.2.12 `PPUMASK_RED` `#define PPUMASK_RED 0b01000000`

20.46.2.13 `PPUMASK_GREEN` `#define PPUMASK_GREEN 0b00100000`

20.46.2.14 PPUMASK_SHOW_SPR `#define PPUMASK_SHOW_SPR 0b00010000`

20.46.2.15 PPUMASK_SHOW_BG `#define PPUMASK_SHOW_BG 0b00001000`

20.46.2.16 PPUMASK_SHOW_SPR_LC `#define PPUMASK_SHOW_SPR_LC 0b00000100`

20.46.2.17 PPUMASK_SHOW_BG_LC `#define PPUMASK_SHOW_BG_LC 0b00000010`

20.46.2.18 PPUMASK_MONOCHROME `#define PPUMASK_MONOCHROME 0b00000001`

20.46.2.19 PPUSTATUS `#define PPUSTATUS ((uint8_t*)0x2002);`

20.46.2.20 OAMADDR `#define OAMADDR ((uint8_t*)0x2003);`

20.46.2.21 OAMDATA `#define OAMDATA ((uint8_t*)0x2004);`

20.46.2.22 PPUSCROLL `#define PPUSCROLL ((uint8_t*)0x2005);`

20.46.2.23 PPUADDR `#define PPUADDR ((uint8_t*)0x2006);`

20.46.2.24 PPUDATA `#define PPUDATA ((uint8_t*)0x2007);`

20.46.2.25 OAMDMA `#define OAMDMA ((uint8_t*)0x4014);`

20.46.2.26 DEVICE_SCREEN_X_OFFSET `#define DEVICE_SCREEN_X_OFFSET 0`

20.46.2.27 DEVICE_SCREEN_Y_OFFSET `#define DEVICE_SCREEN_Y_OFFSET 0`

20.46.2.28 DEVICE_SCREEN_WIDTH `#define DEVICE_SCREEN_WIDTH 32`

20.46.2.29 DEVICE_SCREEN_HEIGHT `#define DEVICE_SCREEN_HEIGHT 30`

20.46.2.30 DEVICE_SCREEN_BUFFER_WIDTH `#define DEVICE_SCREEN_BUFFER_WIDTH 32`

20.46.2.31 DEVICE_SCREEN_BUFFER_HEIGHT `#define DEVICE_SCREEN_BUFFER_HEIGHT 30`

20.46.2.32 **DEVICE_SCREEN_MAP_ENTRY_SIZE** `#define DEVICE_SCREEN_MAP_ENTRY_SIZE 2`

20.46.2.33 **DEVICE_SPRITE_PX_OFFSET_X** `#define DEVICE_SPRITE_PX_OFFSET_X 0`

20.46.2.34 **DEVICE_SPRITE_PX_OFFSET_Y** `#define DEVICE_SPRITE_PX_OFFSET_Y -1`

20.46.2.35 **DEVICE_WINDOW_PX_OFFSET_X** `#define DEVICE_WINDOW_PX_OFFSET_X 0`

20.46.2.36 **DEVICE_WINDOW_PX_OFFSET_Y** `#define DEVICE_WINDOW_PX_OFFSET_Y 0`

20.46.2.37 **DEVICE_SCREEN_PX_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

20.46.2.38 **DEVICE_SCREEN_PX_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

20.46.3 Variable Documentation

20.46.3.1 **shadow_PPUCTRL** `volatile UBYTE shadow_PPUCTRL`

20.46.3.2 **shadow_PPUMASK** `volatile UBYTE shadow_PPUMASK`

20.46.3.3 **bkg_scroll_x** `volatile UBYTE bkg_scroll_x`

20.46.3.4 **bkg_scroll_y** `volatile UBYTE bkg_scroll_y`

20.47 gbdk-lib/include/sms/hardware.h File Reference

`#include <types.h>`

Macros

- `#define __BYTES extern UBYTE`
- `#define __BYTE_REG extern volatile UBYTE`
- `#define MEMCTL_JOYON 0b00000000`
- `#define MEMCTL_JOYOFF 0b00000100`
- `#define MEMCTL_BASEON 0b00000000`
- `#define MEMCTL_BASEOFF 0b00001000`
- `#define MEMCTL_RAMON 0b00000000`
- `#define MEMCTL_RAMOFF 0b00010000`
- `#define MEMCTL_CROMON 0b00000000`
- `#define MEMCTL_CROMOFF 0b00100000`
- `#define MEMCTL_ROMON 0b00000000`

- #define [MEMCTL_ROMOFF](#) 0b01000000
- #define [MEMCTL_EXTON](#) 0b00000000
- #define [MEMCTL_EXTOFF](#) 0b10000000
- #define [JOY_P1_LATCH](#) 0b00000010
- #define [JOY_P2_LATCH](#) 0b00001000
- #define [PSG_LATCH](#) 0x80
- #define [PSG_CH0](#) 0b00000000
- #define [PSG_CH1](#) 0b00100000
- #define [PSG_CH2](#) 0b01000000
- #define [PSG_CH3](#) 0b01100000
- #define [PSG_VOLUME](#) 0b00010000
- #define [STATF_INT_VBL](#) 0b10000000
- #define [STATF_9_SPR](#) 0b01000000
- #define [STATF_SPR_COLL](#) 0b00100000
- #define [VDP_REG_MASK](#) 0b10000000
- #define [VDP_R0](#) 0b10000000
- #define [R0_VSCRL](#) 0b00000000
- #define [R0_VSCRL_INH](#) 0b10000000
- #define [R0_HSCRL](#) 0b00000000
- #define [R0_HSCRL_INH](#) 0b01000000
- #define [R0_NO_LCB](#) 0b00000000
- #define [R0_LCB](#) 0b00100000
- #define [R0_IE1_OFF](#) 0b00000000
- #define [R0_IE1](#) 0b00010000
- #define [R0_SS_OFF](#) 0b00000000
- #define [R0_SS](#) 0b00001000
- #define [R0_DEFAULT](#) 0b00000110
- #define [R0_ES_OFF](#) 0b00000000
- #define [R0_ES](#) 0b00000001
- #define [VDP_R1](#) 0b10000001
- #define [R1_DEFAULT](#) 0b10000000
- #define [R1_DISP_OFF](#) 0b00000000
- #define [R1_DISP_ON](#) 0b01000000
- #define [R1_IE_OFF](#) 0b00000000
- #define [R1_IE](#) 0b00100000
- #define [R1_SPR_8X8](#) 0b00000000
- #define [R1_SPR_8X16](#) 0b00000010
- #define [VDP_R2](#) 0b10000010
- #define [R2_MAP_0x3800](#) 0xFF
- #define [R2_MAP_0x3000](#) 0xFD
- #define [R2_MAP_0x2800](#) 0xFB
- #define [R2_MAP_0x2000](#) 0xF9
- #define [R2_MAP_0x1800](#) 0xF7
- #define [R2_MAP_0x1000](#) 0xF5
- #define [R2_MAP_0x0800](#) 0xF3
- #define [R2_MAP_0x0000](#) 0xF1
- #define [VDP_R3](#) 0b10000011
- #define [VDP_R4](#) 0b10000100
- #define [VDP_R5](#) 0b10000101
- #define [R5_SAT_0x3F00](#) 0xFF
- #define [R5_SAT_MASK](#) 0b10000001
- #define [VDP_R6](#) 0b10000110
- #define [R6_BANK0](#) 0xFB
- #define [R6_DATA_0x0000](#) 0xFB
- #define [R6_BANK1](#) 0xFF

- `#define R6_DATA_0x2000 0xFF`
- `#define VDP_R7 0b10000111`
- `#define VDP_RBORDER 0b10000111`
- `#define R7_COLOR_MASK 0b11110000`
- `#define VDP_R8 0b10001000`
- `#define VDP_RSCX 0b10001000`
- `#define VDP_R9 0b10001001`
- `#define VDP_RSCY 0b10001001`
- `#define VDP_R10 0b10001010`
- `#define R10_INT_OFF 0xFF`
- `#define R10_INT_EVERY 0x00`
- `#define JOY_P1_UP 0b00000001`
- `#define JOY_P1_DOWN 0b00000010`
- `#define JOY_P1_LEFT 0b00000100`
- `#define JOY_P1_RIGHT 0b00001000`
- `#define JOY_P1_SW1 0b00010000`
- `#define JOY_P1_TRIGGER 0b00010000`
- `#define JOY_P1_SW2 0b00100000`
- `#define JOY_P2_UP 0b01000000`
- `#define JOY_P2_DOWN 0b10000000`
- `#define JOY_P2_LEFT 0b00000001`
- `#define JOY_P2_RIGHT 0b00000010`
- `#define JOY_P2_SW1 0b00000100`
- `#define JOY_P2_TRIGGER 0b00000100`
- `#define JOY_P2_SW2 0b00001000`
- `#define JOY_RESET 0b00010000`
- `#define JOY_P1_LIGHT 0b01000000`
- `#define JOY_P2_LIGHT 0b10000000`
- `#define RAMCTL_BANK 0b00000100`
- `#define RAMCTL_ROM 0b00000000`
- `#define RAMCTL_RAM 0b00001000`
- `#define RAMCTL_RO 0b00010000`
- `#define RAMCTL_PROT 0b10000000`
- `#define SYSTEM_PAL 0x00`
- `#define SYSTEM_NTSC 0x01`
- `#define VBK_TILES 0`
- `#define VBK_ATTRIBUTES 1`
- `#define VDP_SAT_TERM 0xD0`
- `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`
- `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

Variables

- `UBYTE shadow_VDP_R0`
- `UBYTE shadow_VDP_R1`
- `UBYTE shadow_VDP_R2`
- `UBYTE shadow_VDP_R3`
- `UBYTE shadow_VDP_R4`
- `UBYTE shadow_VDP_R5`
- `UBYTE shadow_VDP_R6`
- `UBYTE shadow_VDP_R7`
- `UBYTE shadow_VDP_RBORDER`
- `UBYTE shadow_VDP_R8`
- `UBYTE shadow_VDP_RSCX`
- `UBYTE shadow_VDP_R9`

- [UBYTE shadow_VDP_RSCY](#)
- [UBYTE shadow_VDP_R10](#)
- const [UBYTE _BIOS](#)
- const [UBYTE _SYSTEM](#)
- volatile [UBYTE VDP_ATTR_SHIFT](#)

20.47.1 Detailed Description

Defines that let the SMS/GG hardware registers be accessed from C.

20.47.2 Macro Definition Documentation

20.47.2.1 `__BYTES` `#define __BYTES extern UBYTE`

20.47.2.2 `__BYTE_REG` `#define __BYTE_REG extern volatile UBYTE`

20.47.2.3 `MEMCTL_JOYON` `#define MEMCTL_JOYON 0b00000000`

20.47.2.4 `MEMCTL_JOYOFF` `#define MEMCTL_JOYOFF 0b00000100`

20.47.2.5 `MEMCTL_BASEON` `#define MEMCTL_BASEON 0b00000000`

20.47.2.6 `MEMCTL_BASEOFF` `#define MEMCTL_BASEOFF 0b00001000`

20.47.2.7 `MEMCTL_RAMON` `#define MEMCTL_RAMON 0b00000000`

20.47.2.8 `MEMCTL_RAMOFF` `#define MEMCTL_RAMOFF 0b00010000`

20.47.2.9 `MEMCTL_CROMON` `#define MEMCTL_CROMON 0b00000000`

20.47.2.10 `MEMCTL_CROMOFF` `#define MEMCTL_CROMOFF 0b00100000`

20.47.2.11 `MEMCTL_ROMON` `#define MEMCTL_ROMON 0b00000000`

20.47.2.12 `MEMCTL_ROMOFF` `#define MEMCTL_ROMOFF 0b01000000`

20.47.2.13 `MEMCTL_EXTON` `#define MEMCTL_EXTON 0b00000000`

20.47.2.14 `MEMCTL_EXTOFF` `#define MEMCTL_EXTOFF 0b10000000`

20.47.2.15 JOY_P1_LATCH #define JOY_P1_LATCH 0b00000010

20.47.2.16 JOY_P2_LATCH #define JOY_P2_LATCH 0b00001000

20.47.2.17 PSG_LATCH #define PSG_LATCH 0x80

20.47.2.18 PSG_CH0 #define PSG_CH0 0b00000000

20.47.2.19 PSG_CH1 #define PSG_CH1 0b00100000

20.47.2.20 PSG_CH2 #define PSG_CH2 0b01000000

20.47.2.21 PSG_CH3 #define PSG_CH3 0b01100000

20.47.2.22 PSG_VOLUME #define PSG_VOLUME 0b00010000

20.47.2.23 STATF_INT_VBL #define STATF_INT_VBL 0b10000000

20.47.2.24 STATF_9_SPR #define STATF_9_SPR 0b01000000

20.47.2.25 STATF_SPR_COLL #define STATF_SPR_COLL 0b00100000

20.47.2.26 VDP_REG_MASK #define VDP_REG_MASK 0b10000000

20.47.2.27 VDP_R0 #define VDP_R0 0b10000000

20.47.2.28 R0_VSCRL #define R0_VSCRL 0b00000000

20.47.2.29 R0_VSCRL_INH #define R0_VSCRL_INH 0b10000000

20.47.2.30 R0_HSCRL #define R0_HSCRL 0b00000000

20.47.2.31 R0_HSCRL_INH #define R0_HSCRL_INH 0b01000000

20.47.2.32 R0_NO_LCB #define R0_NO_LCB 0b00000000

20.47.2.33 R0_LCB `#define R0_LCB 0b00100000`

20.47.2.34 R0_IE1_OFF `#define R0_IE1_OFF 0b00000000`

20.47.2.35 R0_IE1 `#define R0_IE1 0b00010000`

20.47.2.36 R0_SS_OFF `#define R0_SS_OFF 0b00000000`

20.47.2.37 R0_SS `#define R0_SS 0b00001000`

20.47.2.38 R0_DEFAULT `#define R0_DEFAULT 0b00000110`

20.47.2.39 R0_ES_OFF `#define R0_ES_OFF 0b00000000`

20.47.2.40 R0_ES `#define R0_ES 0b00000001`

20.47.2.41 VDP_R1 `#define VDP_R1 0b10000001`

20.47.2.42 R1_DEFAULT `#define R1_DEFAULT 0b10000000`

20.47.2.43 R1_DISP_OFF `#define R1_DISP_OFF 0b00000000`

20.47.2.44 R1_DISP_ON `#define R1_DISP_ON 0b01000000`

20.47.2.45 R1_IE_OFF `#define R1_IE_OFF 0b00000000`

20.47.2.46 R1_IE `#define R1_IE 0b00100000`

20.47.2.47 R1_SPR_8X8 `#define R1_SPR_8X8 0b00000000`

20.47.2.48 R1_SPR_8X16 `#define R1_SPR_8X16 0b00000010`

20.47.2.49 VDP_R2 `#define VDP_R2 0b10000010`

20.47.2.50 R2_MAP_0x3800 `#define R2_MAP_0x3800 0xFF`

20.47.2.51 R2_MAP_0x3000 #define R2_MAP_0x3000 0xFD

20.47.2.52 R2_MAP_0x2800 #define R2_MAP_0x2800 0xFB

20.47.2.53 R2_MAP_0x2000 #define R2_MAP_0x2000 0xF9

20.47.2.54 R2_MAP_0x1800 #define R2_MAP_0x1800 0xF7

20.47.2.55 R2_MAP_0x1000 #define R2_MAP_0x1000 0xF5

20.47.2.56 R2_MAP_0x0800 #define R2_MAP_0x0800 0xF3

20.47.2.57 R2_MAP_0x0000 #define R2_MAP_0x0000 0xF1

20.47.2.58 VDP_R3 #define VDP_R3 0b10000011

20.47.2.59 VDP_R4 #define VDP_R4 0b10000100

20.47.2.60 VDP_R5 #define VDP_R5 0b10000101

20.47.2.61 R5_SAT_0x3F00 #define R5_SAT_0x3F00 0xFF

20.47.2.62 R5_SAT_MASK #define R5_SAT_MASK 0b10000001

20.47.2.63 VDP_R6 #define VDP_R6 0b10000110

20.47.2.64 R6_BANK0 #define R6_BANK0 0xFB

20.47.2.65 R6_DATA_0x0000 #define R6_DATA_0x0000 0xFB

20.47.2.66 R6_BANK1 #define R6_BANK1 0xFF

20.47.2.67 R6_DATA_0x2000 #define R6_DATA_0x2000 0xFF

20.47.2.68 VDP_R7 #define VDP_R7 0b10000111

20.47.2.69 VDP_RBORDER `#define VDP_RBORDER 0b10000111`

20.47.2.70 R7_COLOR_MASK `#define R7_COLOR_MASK 0b11110000`

20.47.2.71 VDP_R8 `#define VDP_R8 0b10001000`

20.47.2.72 VDP_RSCX `#define VDP_RSCX 0b10001000`

20.47.2.73 VDP_R9 `#define VDP_R9 0b10001001`

20.47.2.74 VDP_RSCY `#define VDP_RSCY 0b10001001`

20.47.2.75 VDP_R10 `#define VDP_R10 0b10001010`

20.47.2.76 R10_INT_OFF `#define R10_INT_OFF 0xFF`

20.47.2.77 R10_INT EVERY `#define R10_INT EVERY 0x00`

20.47.2.78 JOY_P1_UP `#define JOY_P1_UP 0b00000001`

20.47.2.79 JOY_P1_DOWN `#define JOY_P1_DOWN 0b00000010`

20.47.2.80 JOY_P1_LEFT `#define JOY_P1_LEFT 0b00000100`

20.47.2.81 JOY_P1_RIGHT `#define JOY_P1_RIGHT 0b00001000`

20.47.2.82 JOY_P1_SW1 `#define JOY_P1_SW1 0b00010000`

20.47.2.83 JOY_P1_TRIGGER `#define JOY_P1_TRIGGER 0b00010000`

20.47.2.84 JOY_P1_SW2 `#define JOY_P1_SW2 0b00100000`

20.47.2.85 JOY_P2_UP `#define JOY_P2_UP 0b01000000`

20.47.2.86 JOY_P2_DOWN `#define JOY_P2_DOWN 0b10000000`

20.47.2.87 JOY_P2_LEFT #define JOY_P2_LEFT 0b00000001

20.47.2.88 JOY_P2_RIGHT #define JOY_P2_RIGHT 0b00000010

20.47.2.89 JOY_P2_SW1 #define JOY_P2_SW1 0b00000100

20.47.2.90 JOY_P2_TRIGGER #define JOY_P2_TRIGGER 0b00000100

20.47.2.91 JOY_P2_SW2 #define JOY_P2_SW2 0b00001000

20.47.2.92 JOY_RESET #define JOY_RESET 0b00010000

20.47.2.93 JOY_P1_LIGHT #define JOY_P1_LIGHT 0b01000000

20.47.2.94 JOY_P2_LIGHT #define JOY_P2_LIGHT 0b10000000

20.47.2.95 RAMCTL_BANK #define RAMCTL_BANK 0b00000100

20.47.2.96 RAMCTL_ROM #define RAMCTL_ROM 0b00000000

20.47.2.97 RAMCTL_RAM #define RAMCTL_RAM 0b00001000

20.47.2.98 RAMCTL_RO #define RAMCTL_RO 0b00010000

20.47.2.99 RAMCTL_PROT #define RAMCTL_PROT 0b10000000

20.47.2.100 SYSTEM_PAL #define SYSTEM_PAL 0x00

20.47.2.101 SYSTEM_NTSC #define SYSTEM_NTSC 0x01

20.47.2.102 VBK_TILES #define VBK_TILES 0

20.47.2.103 VBK_ATTRIBUTES #define VBK_ATTRIBUTES 1

20.47.2.104 VDP_SAT_TERM #define VDP_SAT_TERM 0xD0

20.47.2.105 **DEVICE_SCREEN_PX_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

20.47.2.106 **DEVICE_SCREEN_PX_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

20.47.3 Variable Documentation

20.47.3.1 **shadow_VDP_R0** `UBYTE shadow_VDP_R0`

20.47.3.2 **shadow_VDP_R1** `UBYTE shadow_VDP_R1`

20.47.3.3 **shadow_VDP_R2** `UBYTE shadow_VDP_R2`

20.47.3.4 **shadow_VDP_R3** `UBYTE shadow_VDP_R3`

20.47.3.5 **shadow_VDP_R4** `UBYTE shadow_VDP_R4`

20.47.3.6 **shadow_VDP_R5** `UBYTE shadow_VDP_R5`

20.47.3.7 **shadow_VDP_R6** `UBYTE shadow_VDP_R6`

20.47.3.8 **shadow_VDP_R7** `UBYTE shadow_VDP_R7`

20.47.3.9 **shadow_VDP_RBORDER** `UBYTE shadow_VDP_RBORDER`

20.47.3.10 **shadow_VDP_R8** `UBYTE shadow_VDP_R8`

20.47.3.11 **shadow_VDP_RSCX** `UBYTE shadow_VDP_RSCX`

20.47.3.12 **shadow_VDP_R9** `UBYTE shadow_VDP_R9`

20.47.3.13 **shadow_VDP_RSCY** `UBYTE shadow_VDP_RSCY`

20.47.3.14 **shadow_VDP_R10** `UBYTE shadow_VDP_R10`

20.47.3.15 **_BIOS** `const UBYTE _BIOS`

20.47.3.16 `_SYSTEM` `const UBYTE _SYSTEM`

20.47.3.17 `VDP_ATTR_SHIFT` `volatile UBYTE VDP_ATTR_SHIFT`

20.48 gbdk-lib/include/gb/isr.h File Reference

```
#include <stdint.h>
#include <types.h>
```

Data Structures

- struct [isr_vector_t](#)
- struct [isr_nested_vector_t](#)

Macros

- `#define VECTOR_STAT 0x48`
- `#define VECTOR_TIMER 0x50`
- `#define VECTOR_SERIAL 0x58`
- `#define VECTOR_JOYPAD 0x60`
- `#define ISR_VECTOR(ADDR, FUNC) static const isr_vector_t AT((ADDR)) __ISR_ ## ADDR = {0xc3, (void *)&(FUNC)};`
- `#define ISR_NESTED_VECTOR(ADDR, FUNC) static const isr_nested_vector_t AT((ADDR)) __ISR_ ## ADDR = {{0xfb, 0xc3}, (void *)&(FUNC)};`

Typedefs

- `typedef struct isr_vector_t isr_vector_t`
- `typedef struct isr_nested_vector_t isr_nested_vector_t`

20.48.1 Detailed Description

Macros for creating raw interrupt service routines (ISRs) which do not use the default GBDK ISR dispatcher. Handlers installed this way will have less overhead than ones which use the GBDK ISR dispatcher.

20.48.2 Macro Definition Documentation

20.48.2.1 `VECTOR_STAT` `#define VECTOR_STAT 0x48`
Address for the STAT interrupt vector

20.48.2.2 `VECTOR_TIMER` `#define VECTOR_TIMER 0x50`
Address for the TIMER interrupt vector

20.48.2.3 `VECTOR_SERIAL` `#define VECTOR_SERIAL 0x58`
Address for the SERIAL interrupt vector

20.48.2.4 `VECTOR_JOYPAD` `#define VECTOR_JOYPAD 0x60`
Address for the JOYPAD interrupt vector

20.48.2.5 ISR_VECTOR `#define ISR_VECTOR(`

```

    ADDR,
    FUNC ) static const isr\_vector\_t AT((ADDR)) __ISR_ ## ADDR = {0xc3, (void *)&(F←
UNC) };
```

Creates an interrupt vector at the given address for a raw interrupt service routine (which does not use the GBDK ISR dispatcher)

Parameters

<i>ADDR</i>	Address of the interrupt vector, any of: VECTOR_STAT , VECTOR_TIMER , VECTOR_SERIAL , VECTOR_JOYPAD
<i>FUNC</i>	ISR function supplied by the user

This cannot be used with the VBLANK interrupt.

Do not use this in combination with interrupt installers that rely on the default GBDK ISR dispatcher such as [add_TIM\(\)](#), [remove_TIM\(\)](#) (and the same for all other interrupts).

Example:

```

#include <gb/isr.h>
void TimerISR() __critical __interrupt {
    // some ISR code here
}
ISR_VECTOR(VECTOR_TIMER, TimerISR)
```

See also

[ISR_NESTED_VECTOR](#), [set_interrupts](#)

20.48.2.6 ISR_NESTED_VECTOR `#define ISR_NESTED_VECTOR(`

```

    ADDR,
    FUNC ) static const isr\_nested\_vector\_t AT((ADDR)) __ISR_ ## ADDR = {{0xfb,
0xc3}}, (void *)&(FUNC)};
```

Creates an interrupt vector at the given address for a raw interrupt service routine allowing nested interrupts

Parameters

<i>ADDR</i>	Address of the interrupt vector, any of: VECTOR_STAT , VECTOR_TIMER , VECTOR_SERIAL , VECTOR_JOYPAD
<i>FUNC</i>	ISR function

This cannot be used with the VBLANK interrupt

The LCD STAT vector ([VECTOR_STAT](#)) cannot be used in the same program as `stdio.h` since they install an ISR vector to the same location.

See also

[ISR_VECTOR](#)

20.48.3 Typedef Documentation

20.48.3.1 [isr_vector_t](#) `typedef struct isr_vector_t isr_vector_t`

20.48.3.2 [isr_nested_vector_t](#) `typedef struct isr_nested_vector_t isr_nested_vector_t`

20.49 gbdk-lib/include/gb/metasprites.h File Reference

```
#include <gb/hardware.h>
```



```
#include <types.h>
#include <stdint.h>
```

Data Structures

- struct [metasprite_t](#)

Macros

- #define [metasprite_end](#) -128
- #define [METASPR_ITEM](#)(dy, dx, dt, a) {(dy),(dx),(dt),(a)}
- #define [METASPR_TERM](#) {metasprite_end}

Typedefs

- typedef struct [metasprite_t](#) [metasprite_t](#)

Functions

- void [hide_sprites_range](#) (UINT8 from, UINT8 to) [OLDCALL PRESERVES_REGS](#)(b
- [uint8_t move_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t move_metasprite_vflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t move_metasprite_hflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t move_metasprite_hvflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- void [hide_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_sprite)

Variables

- const void * [__current_metasprite](#)
- [uint8_t __current_base_tile](#)
- [uint8_t __render_shadow_OAM](#)
- void [c](#)

20.49.1 Detailed Description

20.49.2 Metasprite support

A metasprite is a larger sprite made up from a collection of smaller individual hardware sprites. Different frames of the same metasprites can share tile data.

The api supports metasprites in both [SPRITES_8x8](#) and [SPRITES_8x16](#) mode. If 8x16 mode is used then the height of the metasprite must be a multiple of 16.

The origin (pivot) for the metasprite is not required to be in the upper left-hand corner as with regular hardware sprites.

Use the [utility_png2asset](#) tool to convert single or multiple frames of graphics into metasprite structured data for use with the ...metasprite...() functions.

20.49.3 Metasprites composed of variable numbers of sprites

When using png2asset, it's common for the output of different frames to be composed of different numbers of hardware sprites (since it's trying to create each frame as efficiently as possible). Due to that, it's good practice to clear out (hide) unused sprites in the shadow_OAM that have been set by previous frames.

```
// Example:
// Hide rest of the hardware sprites, because amount
// of sprites differ between animation frames.
// (where hiwater == last hardware sprite used + 1)
for (uint8_t i = hiwater; i < 40; i++) shadow_OAM[i].y = 0;
```

20.49.4 Metasprites and sprite properties (including cgb palette)

When the `move_metasprite_*`() functions are called they update all properties for the affected sprites in the Shadow OAM. This means any existing property flags set for a sprite (CGB palette, BG/WIN priority, Tile VRAM Bank) will get overwritten.

How to use sprite property flags with metasprites:

- Metasprite structures can be copied into RAM so their property flags can be modified at runtime.
- The metasprite structures can have the property flags modified before compilation (such as with `-sp <props>` in the [png2asset](#) tool).
- Update properties for the affected sprites after calling a `move_metasprite_*`() function.

The following functions are only available for Game Boy and related clone consoles due to lack of hardware support for sprite flipping in other consoles. See [docs_consoles_supported_list](#)

- [move_metasprite_vflip\(\)](#)
- [move_metasprite_hflip\(\)](#)
- [move_metasprite_hvflip\(\)](#)

20.49.5 Macro Definition Documentation

20.49.5.1 metasprite_end `#define metasprite_end -128`

20.49.5.2 METASPR_ITEM `#define METASPR_ITEM(
 dy,
 dx,
 dt,
 a) { (dy), (dx), (dt), (a) }`

20.49.5.3 METASPR_TERM `#define METASPR_TERM {metasprite_end}`

20.49.6 Typedef Documentation

20.49.6.1 metasprite_t `typedef struct metasprite_t metasprite_t`
 Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(uint8_t) Start tile relative to the metasprites own set of tiles
<i>props</i>	(uint8_t) Property Flags

Metasprites are built from multiple [metasprite_t](#) items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of [metasprite_t](#) items (which may vary based on how many sprites are required for that particular frame). A metasprite frame is terminated with a `{metasprite_end}` entry.

20.49.7 Function Documentation

20.49.7.1 `hide_sprites_range()` `void hide_sprites_range (`
 `UINT8 from,`
 `UINT8 to)`

Hides all hardware sprites in range from $\leq X < to$

Parameters

<i>from</i>	start OAM index
<i>to</i>	finish OAM index

See also

[hide_sprite](#)

20.49.7.2 `move metasprite()` `uint8_t move_metasprite (`
 `const metasprite_t * metasprite,`
 `uint8_t base_tile,`
 `uint8_t base_sprite,`
 `uint8_t x,`
 `uint8_t y) [inline]`

Moves metasprite to the absolute position **x** and **y**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Moves **metasprite** to the absolute position **x** and **y** (with **no flip** on the X or Y axis). Hardware sprites are allocated starting from **base_sprite**, using tiles starting from **base_tile**.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as CGB Palette), see [Metasprites and sprite properties](#).

Returns

Number of hardware sprites used to draw this metasprite

20.49.7.3 `move metasprite_vflip()` `uint8_t move_metasprite_vflip (`
 `const metasprite_t * metasprite,`
 `uint8_t base_tile,`
 `uint8_t base_sprite,`
 `uint8_t x,`
 `uint8_t y) [inline]`

Moves metasprite to the absolute position **x** and **y**, **flipped on the Y axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move_metasprite\(\)](#), but with the metasprite flipped on the Y axis only.
Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as CGB palette), see [Metasprites and sprite properties](#).
This function is only available on Game Boy and related clone consoles.

Returns

Number of hardware sprites used to draw this metasprite

See also

[move_metasprite\(\)](#)

20.49.7.4 move_metasprite_hflip() `uint8_t move_metasprite_hflip (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move_metasprite\(\)](#), but with the metasprite flipped on the X axis only.
Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as CGB palette), see [Metasprites and sprite properties](#).
This function is only available on Game Boy and related clone consoles.

Returns

Number of hardware sprites used to draw this metasprite

See also

[move metasprite\(\)](#)

20.49.7.5 move metasprite_hvflip() `uint8_t move metasprite_hvflip (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X and Y axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move metasprite\(\)](#), but with the metasprite flipped on both the X and Y axis.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as CGB palette), see [Metasprites and sprite properties](#).
 This function is only available on Game Boy and related clone consoles.

Returns

Number of hardware sprites used to draw this metasprite

See also

[move metasprite\(\)](#)

20.49.7.6 hide metasprite() `void hide metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_sprite) [inline]`

Hides a metasprite from the screen

Parameters

<i>metasprite</i>	Pointer to first struct of the desired metasprite frame
<i>base_sprite</i>	Number of hardware sprite to start with

Sets:

- `__current_metasprite = metasprite;`

20.49.8 Variable Documentation

20.49.8.1 `__current_metasprite` `const void* __current_metasprite`

20.49.8.2 `__current_base_tile` `uint8_t __current_base_tile`

20.49.8.3 `__render_shadow_OAM` `uint8_t __render_shadow_OAM`

20.49.8.4 `c` `void c`

20.50 gbdk-lib/include/gbdk/metasprites.h File Reference

```
#include <gb/metasprites.h>
```

20.51 gbdk-lib/include/msx/metasprites.h File Reference

```
#include <msx/hardware.h>
#include <types.h>
#include <stdint.h>
```

Data Structures

- struct [metasprite_t](#)

Macros

- #define [metasprite_end](#) -128
- #define [METASPR_ITEM](#)(dy, dx, dt, a) {(dy),(dx),(dt),(a)}
- #define [METASPR_TERM](#) {[metasprite_end](#)}

Typedefs

- typedef struct [metasprite_t](#) [metasprite_t](#)

Functions

- void [hide_sprites_range](#) (UINT8 from, UINT8 to) [Z88DK_CALLEE PRESERVES_REGS](#)(iyh
- [uint8_t move_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- void [hide_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_sprite)

Variables

- const void * [__current_metasprite](#)
- [uint8_t __current_base_tile](#)
- [uint8_t __render_shadow_OAM](#)
- static [uint8_t iyl](#)

20.51.1 Macro Definition Documentation

20.51.1.1 `metasprite_end` `#define metasprite_end -128`

20.51.1.2 METASPR_ITEM `#define METASPR_ITEM(`
`dy,`
`dx,`
`dt,`
`a) { (dy), (dx), (dt), (a) }`

20.51.1.3 METASPR_TERM `#define METASPR_TERM {metasprite_end}`

20.51.2 Typedef Documentation

20.51.2.1 metasprite_t `typedef struct metasprite_t metasprite_t`
Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtil</i>	(uint8_t) Start tile relative to the metasprites own set of tiles
<i>props</i>	(uint8_t) Property Flags

Metasprites are built from multiple [metasprite_t](#) items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of [metasprite_t](#) items (which may vary based on how many sprites are required for that particular frame). A metasprite frame is terminated with a {metasprite_end} entry.

20.51.3 Function Documentation

20.51.3.1 hide_sprites_range() `void hide_sprites_range (`
`UINT8 from,`
`UINT8 to)`

Hides all hardware sprites in range from $\leq X < to$

Parameters

<i>from</i>	start OAM index
<i>to</i>	finish OAM index

20.51.3.2 move_metasprite() `uint8_t move_metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position x and y

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start

Parameters

<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Moves **metasprite** to the absolute position **x** and **y** (with **no flip** on the X or Y axis). Hardware sprites are allocated starting from **base_sprite**, using tiles starting from **base_tile**.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Returns

Number of hardware sprites used to draw this metasprite

20.51.3.3 hide_metasprite() `void hide_metasprite (`
 `const metasprite_t * metasprite,`
 `uint8_t base_sprite) [inline]`

Hides a metasprite from the screen

Parameters

<i>metasprite</i>	Pointer to first struct of the desired metasprite frame
<i>base_sprite</i>	Number of hardware sprite to start with

Sets:

- `__current_metasprite = metasprite;`

20.51.4 Variable Documentation

20.51.4.1 __current_metasprite `const void* __current_metasprite`

20.51.4.2 __current_base_tile `uint8_t __current_base_tile`

20.51.4.3 __render_shadow_OAM `uint8_t __render_shadow_OAM`

20.51.4.4 iyl `uint8_t iyl`

20.52 gbdk-lib/include/nes/metaspprites.h File Reference

```
#include <nes/hardware.h>
#include <types.h>
#include <stdint.h>
```


Data Structures

- struct [metasprite_t](#)

Macros

- #define [metasprite_end](#) -128
- #define [METASPR_ITEM](#)(dy, dx, dt, a) {(dy),(dx),(dt),(a)}
- #define [METASPR_TERM](#) {metasprite_end}

Typedefs

- typedef struct [metasprite_t](#) [metasprite_t](#)

Functions

- void [hide_sprites_range](#) (UINT8 from, UINT8 to) [OLDCALL](#)
- [uint8_t](#) [move_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t](#) [move_metasprite_vflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t](#) [move_metasprite_hflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- [uint8_t](#) [move_metasprite_hvflip](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_tile, [uint8_t](#) base_sprite, [uint8_t](#) x, [uint8_t](#) y)
- void [hide_metasprite](#) (const [metasprite_t](#) *metasprite, [uint8_t](#) base_sprite)

Variables

- const void * [__current_metasprite](#)
- [uint8_t](#) [__current_base_tile](#)
- [uint8_t](#) [__render_shadow_OAM](#)

20.52.1 Detailed Description

20.52.2 Metasprite support

A metasprite is a larger sprite made up from a collection of smaller individual hardware sprites. Different frames of the same metasprites can share tile data.

The api supports metasprites in both [SPRITES_8x8](#) and [SPRITES_8x16](#) mode. If 8x16 mode is used then the height of the metasprite must be a multiple of 16.

The origin (pivot) for the metasprite is not required to be in the upper left-hand corner as with regular hardware sprites.

Use the [utility_png2asset](#) tool to convert single or multiple frames of graphics into metasprite structured data for use with the ...metasprite...() functions.

20.52.3 Metasprites composed of variable numbers of sprites

When using png2asset, it's common for the output of different frames to be composed of different numbers of hardware sprites (since it's trying to create each frame as efficiently as possible). Due to that, it's good practice to clear out (hide) unused sprites in the shadow_OAM that have been set by previous frames.

```
// Example:
// Hide rest of the hardware sprites, because amount
// of sprites differ between animation frames.
// (where hiwater == last hardware sprite used + 1)
for (uint8\_t i = hiwater; i < 64; i++) shadow\_OAM[i].y = 0;
```

20.52.4 Metasprites and sprite properties (including cgb palette)

When the `move_metasprite_*`() functions are called they update all properties for the affected sprites in the Shadow OAM. This means any existing property flags set for a sprite will get overwritten.

How to use sprite property flags with metasprites:

- Metasprite structures can be copied into RAM so their property flags can be modified at runtime.
- The metasprite structures can have the property flags modified before compilation (such as with `-sp <props>` in the [png2asset](#) tool).
- Update properties for the affected sprites after calling a `move_metasprite_*`() function.

20.52.5 Macro Definition Documentation

20.52.5.1 `metasprite_end` `#define metasprite_end -128`

20.52.5.2 `METASPR_ITEM` `#define METASPR_ITEM(
 dy,
 dx,
 dt,
 a) { (dy), (dx), (dt), (a) }`

20.52.5.3 `METASPR_TERM` `#define METASPR_TERM {metasprite_end}`

20.52.6 Typedef Documentation

20.52.6.1 `metasprite_t` `typedef struct metasprite_t metasprite_t`
Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(uint8_t) Start tile relative to the metasprites own set of tiles
<i>props</i>	(uint8_t) Property Flags

Metasprites are built from multiple [metasprite_t](#) items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of [metasprite_t](#) items (which may vary based on how many sprites are required for that particular frame).

A metasprite frame is terminated with a `{metasprite_end}` entry.

20.52.7 Function Documentation

20.52.7.1 `hide_sprites_range()` `void hide_sprites_range (
 UINT8 from,
 UINT8 to)`

Hides all hardware sprites in range from `<= X < to`

Parameters

<i>from</i>	start OAM index
<i>to</i>	finish OAM index

Hides all hardware sprites in range from $\leq X < to$

Parameters

<i>from</i>	start OAM index
<i>to</i>	finish OAM index

See also

[hide_sprite](#)

20.52.7.2 move metasprite() `uint8_t move metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position **x** and **y**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Moves **metasprite** to the absolute position **x** and **y** (with **no flip** on the X or Y axis). Hardware sprites are allocated starting from **base_sprite**, using tiles starting from **base_tile**.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as palette), see [Metasprites and sprite properties](#).

Returns

Number of hardware sprites used to draw this metasprite

20.52.7.3 move metasprite_vflip() `uint8_t move metasprite_vflip (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position **x** and **y**, **flipped on the Y axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move_metasprite\(\)](#), but with the metasprite flipped on the Y axis only.
Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as palette), see [Metaspprites and sprite properties](#).

Returns

Number of hardware sprites used to draw this metasprite

See also

[move_metasprite\(\)](#)

20.52.7.4 move_metasprite_hflip() `uint8_t move_metasprite_hflip (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move_metasprite\(\)](#), but with the metasprite flipped on the X axis only.
Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as palette), see [Metaspprites and sprite properties](#).

Returns

Number of hardware sprites used to draw this metasprite

See also

[move_metasprite\(\)](#)

20.52.7.5 move_metasprite_hvflip() `uint8_t move_metasprite_hvflip (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X and Y axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move_metasprite\(\)](#), but with the metasprite flipped on both the X and Y axis.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as palette), see [Metasprites and sprite properties](#).

Returns

Number of hardware sprites used to draw this metasprite

See also

[move_metasprite\(\)](#)

20.52.7.6 hide_metasprite() `void hide_metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_sprite) [inline]`

Hides a metasprite from the screen

Parameters

<i>metasprite</i>	Pointer to first struct of the desired metasprite frame
<i>base_sprite</i>	Number of hardware sprite to start with

Sets:

- `__current_metasprite = metasprite;`

20.52.8 Variable Documentation

20.52.8.1 __current_metasprite `const void* __current_metasprite`

20.52.8.2 __current_base_tile `uint8_t __current_base_tile`

20.52.8.3 `__render_shadow_OAM` `uint8_t __render_shadow_OAM`**20.53 gbdk-lib/include/sms/metaspprites.h File Reference**

```
#include <sms/hardware.h>
#include <types.h>
#include <stdint.h>
```

Data Structures

- struct `metasprite_t`

Macros

- #define `metasprite_end` -128
- #define `METASPR_ITEM`(dy, dx, dt, a) {(dy),(dx),(dt)}
- #define `METASPR_TERM` {`metasprite_end`}

Typedefs

- typedef struct `metasprite_t` `metasprite_t`

Functions

- void `hide_sprites_range` (UINT8 from, UINT8 to) `Z88DK_CALLEE PRESERVES_REGS`(iyh
- `uint8_t` `move_metasprite` (const `metasprite_t` *metasprite, `uint8_t` base_tile, `uint8_t` base_sprite, `uint8_t` x, `uint8_t` y)
- void `hide_metasprite` (const `metasprite_t` *metasprite, `uint8_t` base_sprite)

Variables

- const void * `__current_metasprite`
- `uint8_t` `__current_base_tile`
- `uint8_t` `__render_shadow_OAM`
- static `uint8_t` `iyi`

20.53.1 Detailed Description**20.53.2 Metasprite support**

A metasprite is a larger sprite made up from a collection of smaller individual hardware sprites. Different frames of the same metasprites can share tile data.

The api supports metasprites in both `SPRITES_8x8` and `SPRITES_8x16` mode. If 8x16 mode is used then the height of the metasprite must be a multiple of 16.

The origin (pivot) for the metasprite is not required to be in the upper left-hand corner as with regular hardware sprites.

Use the `utility_png2asset` tool to convert single or multiple frames of graphics into metasprite structured data for use with the `...metasprite...`() functions.

20.53.3 Metaspprites composed of variable numbers of sprites

When using `png2asset`, it's common for the output of different frames to be composed of different numbers of hardware sprites (since it's trying to create each frame as efficiently as possible). Due to that, it's good practice to clear out (hide) unused sprites in the `shadow_OAM` that have been set by previous frames.

20.53.4 Metasprite support

A metasprite is a larger sprite made up from a collection of smaller individual hardware sprites. Different frames of the same metasprites can share tile data.

The api supports metasprites in both `SPRITES_8x8` and `SPRITES_8x16` mode. If 8x16 mode is used then the height of the metasprite must be a multiple of 16.

The origin (pivot) for the metasprite is not required to be in the upper left-hand corner as with regular hardware sprites.

Use the `utility_png2asset` tool to convert single or multiple frames of graphics into metasprite structured data for use with the `...metasprite...`() functions.

20.53.5 Metasprites composed of variable numbers of sprites

When using `png2asset`, it's common for the output of different frames to be composed of different numbers of hardware sprites (since it's trying to create each frame as efficiently as possible). Due to that, it's good practice to clear out (hide) unused sprites in the `shadow_OAM` that have been set by previous frames.

20.53.6 Macro Definition Documentation

20.53.6.1 `metasprite_end` `#define metasprite_end -128`

20.53.6.2 `METASPR_ITEM` `#define METASPR_ITEM(
 dy,
 dx,
 dt,
 a) { (dy), (dx), (dt) }`

20.53.6.3 `METASPR_TERM` `#define METASPR_TERM {metasprite_end}`

20.53.7 Typedef Documentation

20.53.7.1 `metasprite_t` `typedef struct metasprite_t metasprite_t`
 Metasprite sub-item structure

Parameters

<i>dy</i>	(int8_t) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(int8_t) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtil</i>	(uint8_t) Start tile relative to the metasprites own set of tiles

Metasprites are built from multiple `metasprite_t` items (one for each sub-sprite) and a pool of tiles they reference. If a metasprite has multiple frames then each frame will be built from some number of `metasprite_t` items (which may vary based on how many sprites are required for that particular frame).

A metasprite frame is terminated with a `{metasprite_end}` entry.

20.53.8 Function Documentation

20.53.8.1 `hide_sprites_range()` `void hide_sprites_range (
 UINT8 from,`

`UINT8 to)`

Hides all hardware sprites in range from $\leq X < to$

Parameters

<i>from</i>	start OAM index
<i>to</i>	finish OAM index

20.53.8.2 move_metasprite() `uint8_t move_metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_tile,`
`uint8_t base_sprite,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves metasprite to the absolute position **x** and **y**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Moves **metasprite** to the absolute position **x** and **y** (with **no flip** on the X or Y axis). Hardware sprites are allocated starting from **base_sprite**, using tiles starting from **base_tile**.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Returns

Number of hardware sprites used to draw this metasprite

20.53.8.3 hide_metasprite() `void hide_metasprite (`
`const metasprite_t * metasprite,`
`uint8_t base_sprite) [inline]`

Hides a metasprite from the screen

Parameters

<i>metasprite</i>	Pointer to first struct of the desired metasprite frame
<i>base_sprite</i>	Number of hardware sprite to start with

Sets:

- `__current_metasprite = metasprite;`

20.53.9 Variable Documentation

20.53.9.1 `__current metasprite` `const void* __current_metasprite`

20.53.9.2 `__current_base_tile` `uint8_t __current_base_tile`

20.53.9.3 `__render_shadow_OAM` `uint8_t __render_shadow_OAM`

20.53.9.4 `iy1` `void iy1`

20.54 gbdk-lib/include/gb/sgb.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define SGB_PAL_01` `0x00U`
- `#define SGB_PAL_23` `0x01U`
- `#define SGB_PAL_03` `0x02U`
- `#define SGB_PAL_12` `0x03U`
- `#define SGB_ATTR_BLK` `0x04U`
- `#define SGB_ATTR_LIN` `0x05U`
- `#define SGB_ATTR_DIV` `0x06U`
- `#define SGB_ATTR_CHR` `0x07U`
- `#define SGB_SOUND` `0x08U`
- `#define SGB_SOU_TRN` `0x09U`
- `#define SGB_PAL_SET` `0x0AU`
- `#define SGB_PAL_TRN` `0x0BU`
- `#define SGB_ATTRC_EN` `0x0CU`
- `#define SGB_TEST_EN` `0x0DU`
- `#define SGB_ICON_EN` `0x0EU`
- `#define SGB_DATA_SND` `0x0FU`
- `#define SGB_DATA_TRN` `0x10U`
- `#define SGB_MLT_REQ` `0x11U`
- `#define SGB_JUMP` `0x12U`
- `#define SGB_CHR_TRN` `0x13U`
- `#define SGB_PCT_TRN` `0x14U`
- `#define SGB_ATTR_TRN` `0x15U`
- `#define SGB_ATTR_SET` `0x16U`
- `#define SGB_MASK_EN` `0x17U`
- `#define SGB_OBJ_TRN` `0x18U`

Functions

- `uint8_t sgb_check ()` `OLDCALL PRESERVES_REGS(b`
- `void sgb_transfer (uint8_t *packet)` `OLDCALL PRESERVES_REGS(b`

Variables

- `uint8_t c`

20.54.1 Detailed Description

Super Gameboy definitions.
See the example SGB project for additional details.

20.54.2 Macro Definition Documentation

20.54.2.1 SGB_PAL_01 `#define SGB_PAL_01 0x00U`
SGB Command: Set SGB Palettes 0 & 1

20.54.2.2 SGB_PAL_23 `#define SGB_PAL_23 0x01U`
SGB Command: Set SGB Palettes 2 & 3

20.54.2.3 SGB_PAL_03 `#define SGB_PAL_03 0x02U`
SGB Command: Set SGB Palettes 0 & 3

20.54.2.4 SGB_PAL_12 `#define SGB_PAL_12 0x03U`
SGB Command: Set SGB Palettes 1 & 2

20.54.2.5 SGB_ATTR_BLK `#define SGB_ATTR_BLK 0x04U`
SGB Command: Set color attributes for rectangular regions

20.54.2.6 SGB_ATTR_LIN `#define SGB_ATTR_LIN 0x05U`
SGB Command: Set color attributes for horizontal or vertical character lines

20.54.2.7 SGB_ATTR_DIV `#define SGB_ATTR_DIV 0x06U`
SGB Command: Split screen in half and assign separate color attribes to each side and the divider

20.54.2.8 SGB_ATTR_CHR `#define SGB_ATTR_CHR 0x07U`
SGB Command: Set color attributes for separate charactersSet SGB Palette 0,1 Data

20.54.2.9 SGB_SOUND `#define SGB_SOUND 0x08U`
SGB Command: Start and stop a internal sound effect, and sounds using internal tone data

20.54.2.10 SGB_SOU_TRN `#define SGB_SOU_TRN 0x09U`
SGB Command: Transfer sound code or data to the SNES APU RAM

20.54.2.11 SGB_PAL_SET `#define SGB_PAL_SET 0x0AU`
SGB Command: Apply (previously transferred) SGB system color palettes to actual SNES palettes

20.54.2.12 SGB_PAL_TRN `#define SGB_PAL_TRN 0x0BU`
SGB Command: Transfer palette data into SGB system color palettes

20.54.2.13 SGB_ATTRC_EN `#define SGB_ATTRC_EN 0x0CU`
SGB Command: Enable/disable Attraction mode. It is enabled by default

20.54.2.14 SGB_TEST_EN `#define SGB_TEST_EN 0x0DU`
SGB Command: Enable/disable test mode for "SGB-CPU variable clock speed function"

20.54.2.15 SGB_ICON_EN `#define SGB_ICON_EN 0x0EU`
SGB Command: Enable/disable ICON functionality

20.54.2.16 SGB_DATA_SND `#define SGB_DATA_SND 0x0FU`

SGB Command: Write one or more bytes into SNES Work RAM

20.54.2.17 SGB_DATA_TRN `#define SGB_DATA_TRN 0x10U`

SGB Command: Transfer code or data into SNES RAM

20.54.2.18 SGB_MLT_REQ `#define SGB_MLT_REQ 0x11U`

SGB Command: Request multiplayer mode (input from more than one joypad)

20.54.2.19 SGB_JUMP `#define SGB_JUMP 0x12U`

SGB Command: Set the SNES program counter and NMI (vblank interrupt) handler to specific addresses

20.54.2.20 SGB_CHR_TRN `#define SGB_CHR_TRN 0x13U`

SGB Command: Transfer tile data (characters) to SNES Tile memory

20.54.2.21 SGB_PCT_TRN `#define SGB_PCT_TRN 0x14U`

SGB Command: Transfer tile map and palette data to SNES BG Map memory

20.54.2.22 SGB_ATTR_TRN `#define SGB_ATTR_TRN 0x15U`

SGB Command: Transfer data to (color) Attribute Files (ATFs) in SNES RAM

20.54.2.23 SGB_ATTR_SET `#define SGB_ATTR_SET 0x16U`

SGB Command: Transfer attributes from (color) Attribute Files (ATF) to the Game Boy window

20.54.2.24 SGB_MASK_EN `#define SGB_MASK_EN 0x17U`

SGB Command: Modify Game Boy window mask settings

20.54.2.25 SGB_OBJ_TRN `#define SGB_OBJ_TRN 0x18U`

SGB Command: Transfer OBJ attributes to SNES OAM memory

20.54.3 Function Documentation

20.54.3.1 sgb_check() `uint8_t sgb_check ()`

Returns a non-null value if running on Super GameBoy

20.54.3.2 sgb_transfer() `void sgb_transfer (`
`uint8_t * packet)`

Transfer a SGB packet

Parameters

<code>packet</code>	Pointer to buffer with SGB packet data.
---------------------	---

The first byte of **packet** should be a SGB command, then up to 15 bytes of command parameter data.

See the `sgb_border` GBDK example project for a demo of how to use these the `sgb` functions.

When using the SGB with a PAL SNES, a delay should be added just after program startup such as:

```
// Wait 4 frames
// For PAL SNES this delay is required on startup
for (uint8_t i = 4; i != 0; i--) wait_vbl_done();
```

See also

[sgb_check\(\)](#)

20.54.4 Variable Documentation

20.54.4.1 `c` `void c`

20.55 gbdk-lib/include/gbdk/console.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Functions

- void [gotoxy](#) ([uint8_t](#) x, [uint8_t](#) y) `OLDCALL`
- [uint8_t](#) [posx](#) () `OLDCALL`
- [uint8_t](#) [posy](#) () `OLDCALL`
- void [setchar](#) (char c) `OLDCALL`
- void [cls](#) ()

20.55.1 Detailed Description

Console functions that work like Turbo C's.
The font is 8x8, making the screen 20x18 characters.

20.55.2 Function Documentation

20.55.2.1 [gotoxy\(\)](#) `void gotoxy (` `uint8_t x,` `uint8_t y)`

Move the cursor to an absolute position at **x**, **y**.
x and **y** have units of tiles (8 pixels per unit)

See also

[setchar\(\)](#)

20.55.2.2 [posx\(\)](#) `uint8_t posx ()`

Returns the current X position of the cursor.

See also

[gotoxy\(\)](#)

20.55.2.3 [posy\(\)](#) `uint8_t posy ()`

Returns the current Y position of the cursor.

See also

[gotoxy\(\)](#)

20.55.2.4 setchar() `void setchar (`
 `char c)`

Writes out a single character at the current cursor position.
Does not update the cursor or interpret the character.

See also

[gotoxy\(\)](#)

20.55.2.5 cls() `void cls ()`

Clears the screen

20.56 gbdk-lib/include/gbdk/far_ptr.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Data Structures

- union [__far_ptr](#)

Macros

- `#define TO_FAR_PTR(ofs, seg) (((FAR_PTR)seg << 16) | (FAR_PTR)ofs)`
- `#define FAR_SEG(ptr) (((union __far_ptr *)&ptr)->segofs.seg)`
- `#define FAR_OFS(ptr) (((union __far_ptr *)&ptr)->segofs.ofs)`
- `#define FAR_FUNC(ptr, typ) ((typ)(((union __far_ptr *)&ptr)->segfn.fn))`
- `#define FAR_CALL(ptr, typ, ...) (__call_banked_ptr=ptr,((typ)(&__call_banked))(__VA_ARGS__))`

Typedefs

- `typedef uint32_t FAR_PTR`

Functions

- `void __call_banked ()`
- `uint32_t to_far_ptr (void *ofs, uint16_t seg) OLDCALL`

Variables

- volatile `FAR_PTR __call_banked_ptr`
- volatile `void * __call_banked_addr`
- volatile `uint8_t __call_banked_bank`

20.56.1 Detailed Description

Far pointers include a segment (bank) selector so they are able to point to addresses (functions or data) outside of the current bank (unlike normal pointers which are not bank-aware).
See the `banks_farptr` example project included with gbdk.

Todo Add link to a discussion about banking (such as, how to assign code and variables to banks)

20.56.2 Macro Definition Documentation

20.56.2.1 TO_FAR_PTR `#define TO_FAR_PTR(
 ofs,
 seg) (((FAR_PTR)seg << 16) | (FAR_PTR)ofs)`

Macro to obtain a far pointer at compile-time

Parameters

<i>ofs</i>	Memory address within the given Segment (Bank)
<i>seg</i>	Segment (Bank) number

Returns

A far pointer (type [FAR_PTR](#))

20.56.2.2 FAR_SEG `#define FAR_SEG(
 ptr) (((union __far_ptr *)&ptr)->segofs.seg)`

Macro to get the Segment (Bank) number of a far pointer

Parameters

<i>ptr</i>	A far pointer (type FAR_PTR)
------------	---

Returns

Segment (Bank) of the far pointer (type `uint16_t`)

20.56.2.3 FAR_OFS `#define FAR_OFS(
 ptr) (((union __far_ptr *)&ptr)->segofs.ofs)`

Macro to get the Offset (address) of a far pointer

Parameters

<i>ptr</i>	A far pointer (type FAR_PTR)
------------	---

Returns

Offset (address) of the far pointer (type `void *`)

20.56.2.4 FAR_FUNC `#define FAR_FUNC(
 ptr,
 typ) ((typ) (((union __far_ptr *)&ptr)->segfn.fn))`

20.56.2.5 FAR_CALL `#define FAR_CALL(
 ptr,
 typ,
 ...) (__call_banked_ptr=ptr, ((typ) (&__call_banked)) (__VA_ARGS__))`

Macro to call a function at far pointer **ptr** of type **typ**

Parameters

<i>ptr</i>	Far pointer of a function to call (type FAR_PTR)
<i>typ</i>	Type to cast the function far pointer to.
...	VA Args list of parameters for the function

type should match the definition of the function being called. For example:

```
// A function in bank 2
#pragma bank 2
uint16_t some_function(uint16_t param1, uint16_t param2) __banked { return 1; };
...
// Code elsewhere, such as unbanked main()
// This type declaration should match the above function
typedef uint16_t (*some_function_t)(uint16_t, uint16_t) __banked;
// Using FAR_CALL() with the above as *ptr*, *typ*, and two parameters.
result = FAR_CALL(some_function, some_function_t, 100, 50);
```

Returns

Value returned by the function (if present)

20.56.3 Typedef Documentation

20.56.3.1 [FAR_PTR](#) typedef [uint32_t](#) [FAR_PTR](#)

Type for storing a [FAR_PTR](#)

20.56.4 Function Documentation

20.56.4.1 [__call_banked\(\)](#) void [__call_banked](#) ()20.56.4.2 [to_far_ptr\(\)](#) [uint32_t](#) [to_far_ptr](#) (

void * *ofs*,

[uint16_t](#) *seg*)

Obtain a far pointer at runtime

Parameters

<i>ofs</i>	Memory address within the given Segment (Bank)
<i>seg</i>	Segment (Bank) number

Returns

A far pointer (type [FAR_PTR](#))

20.56.5 Variable Documentation

20.56.5.1 [__call_banked_ptr](#) volatile [FAR_PTR](#) [__call_banked_ptr](#)20.56.5.2 [__call_banked_addr](#) volatile void* [__call_banked_addr](#)20.56.5.3 [__call_banked_bank](#) volatile [uint8_t](#) [__call_banked_bank](#)

20.57 gbdk-lib/include/gbdk/font.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Data Structures

- struct [sfont_handle](#)

Macros

- #define [FONT_256ENCODING](#) 0
- #define [FONT_128ENCODING](#) 1
- #define [FONT_NOENCODING](#) 2
- #define [FONT_COMPRESSED](#) 4

Typedefs

- typedef [uint16_t](#) [font_t](#)
- typedef struct [sfont_handle](#) [mfont_handle](#)
- typedef struct [sfont_handle](#) * [pmfont_handle](#)

Functions

- void [font_init](#) ()
- [font_t](#) [font_load](#) (void *font) [OLDCALL](#)
- [font_t](#) [font_set](#) ([font_t](#) font_handle) [OLDCALL](#)
- void [font_color](#) ([uint8_t](#) forecolor, [uint8_t](#) backcolor) [OLDCALL](#)

Variables

- [uint8_t](#) [font_spect](#) []
- [uint8_t](#) [font_italic](#) []
- [uint8_t](#) [font_ibm](#) []
- [uint8_t](#) [font_min](#) []
- [uint8_t](#) [font_ibm_fixed](#) []

20.57.1 Detailed Description

Multiple font support for the GameBoy Michael Hope, 1999 michaelh@earthling.net

20.57.2 Macro Definition Documentation

20.57.2.1 FONT_256ENCODING `#define FONT_256ENCODING 0`
Various flags in the font header.

20.57.2.2 FONT_128ENCODING `#define FONT_128ENCODING 1`

20.57.2.3 FONT_NOENCODING `#define FONT_NOENCODING 2`

20.57.2.4 FONT_COMPRESSED `#define FONT_COMPRESSED 4`

20.57.3 Typedef Documentation

20.57.3.1 **font_t** `typedef uint16_t font_t`

`font_t` is a handle to a font loaded by [font_load\(\)](#). It can be used with [font_set\(\)](#)

20.57.3.2 **mfont_handle** `typedef struct sfont_handle mfont_handle`

Internal representation of a font. What a `font_t` really is

20.57.3.3 **pmfont_handle** `typedef struct sfont_handle* pmfont_handle`

20.57.4 Function Documentation

20.57.4.1 **font_init()** `void font_init ()`

Initializes the font system. Should be called before other font functions.

20.57.4.2 **font_load()** `font_t font_load (void * font)`

Load a font and set it as the current font.

Parameters

<i>font</i>	Pointer to a font to load (usually a gbdk font)
-------------	---

Returns

Handle to the loaded font, which can be used with [font_set\(\)](#)

See also

[font_init\(\)](#), [font_set\(\)](#), [List of gbdk fonts](#)

20.57.4.3 **font_set()** `font_t font_set (font_t font_handle)`

Set the current font.

Parameters

<i>font_handle</i>	handle of a font returned by font_load()
--------------------	--

Returns

The previously used font handle.

See also

[font_init\(\)](#), [font_load\(\)](#)

20.57.4.4 **font_color()** `void font_color (uint8_t forecolor, uint8_t backcolor)`

Set the current **foreground** colour (for pixels), **background** colour

20.58 gbdk-lib/include/gbdk/gbdk-lib.h File Reference

```
#include <asm/sm83/provides.h>
```

20.58.1 Detailed Description

Settings for the greater library system.

20.59 gbdk-lib/include/gbdk/incbin.h File Reference

```
#include <stdint.h>
```

Macros

- `#define INCBIN_EXTERN(VARNAME)`
- `#define INCBIN_SIZE(VARNAME) ((uint16_t) & __size_ ## VARNAME)`
- `#define BANK(VARNAME) ((uint8_t) & __bank_ ## VARNAME)`
- `#define INCBIN(VARNAME, FILEPATH)`

20.59.1 Detailed Description

Allows binary data from other files to be included into a C source file.

It is implemented using `asm .incbin` and macros.

See the `incbin` example project for a demo of how to use it.

20.59.2 Macro Definition Documentation

20.59.2.1 INCBIN_EXTERN `#define INCBIN_EXTERN(VARNAME)`

Value:

```
extern const uint8_t VARNAME[]; \
extern const void __size_ ## VARNAME; \
extern const void __bank_ ## VARNAME;
```

Creates extern entries for accessing a [INCBIN\(\)](#) generated variable and it's size in another source file.

Parameters

<code>VARNAME</code>	Name of the variable used with <code>INCBIN</code>
----------------------	--

An entry is created for the variable and it's size variable.

[INCBIN\(\)](#), [INCBIN_SIZE\(\)](#)

20.59.2.2 INCBIN_SIZE `#define INCBIN_SIZE(VARNAME) ((uint16_t) & __size_ ## VARNAME)`

Obtains the **size in bytes** of the [INCBIN\(\)](#) generated data

Parameters

<code>VARNAME</code>	Name of the variable used with <code>INCBIN</code>
----------------------	--

Requires [INCBIN_EXTERN\(\)](#) to have been called earlier in the source file

[INCBIN\(\)](#), [INCBIN_EXTERN\(\)](#)

20.59.2.3 BANK `#define BANK(VARNAME) ((uint8_t) & __bank_ ## VARNAME)`

Obtains the **bank number** of the [INCBIN\(\)](#) generated data

Parameters

<i>VARNAME</i>	Name of the variable used with INCBIN
----------------	---------------------------------------

Requires [INCBIN_EXTERN\(\)](#) to have been called earlier in the source file [INCBIN\(\)](#), [INCBIN_EXTERN\(\)](#)

20.59.2.4 INCBIN `#define INCBIN(
 VARNAME,
 FILEPATH)`

Value:

```
void __func__ ## VARNAME() __banked __naked { \
__asm \
_ ## VARNAME:: \
1$: \
.incbin FILEPATH \
2$: \
__size__ ## VARNAME = (2$-1$) \
.globl __size__ ## VARNAME \
.local b__func__ ## VARNAME \
__bank__ ## VARNAME = b__func__ ## VARNAME \
.globl __bank__ ## VARNAME \
__endasm; \
}
```

Includes binary data into a C source file

Parameters

<i>VARNAME</i>	Variable name to use
<i>FILEPATH</i>	Path to the file which will be binary included into the C source file

filepath is relative to the working directory of the tool that is calling it (often a makefile's working directory), **NOT** to the file it's being included into.

The variable name is not modified and can be used as-is.

See also

[INCBIN_SIZE\(\)](#) for obtaining the size of the included data.

[BANK\(\)](#) for obtaining the bank number of the included data.

Use [INCBIN_EXTERN\(\)](#) within another source file to make the variable and it's data accesible there.

20.60 gbdk-lib/include/gbdk/platform.h File Reference

```
#include <gb/gb.h>
#include <gb/cgb.h>
#include <gb/sgb.h>
```

20.61 gbdk-lib/include/gbdk/rledecompress.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define RLE_STOP 0`

Functions

- [uint8_t rle_init](#) (void *data) [OLDCALL](#)
- [uint8_t rle_decompress](#) (void *dest, [uint8_t](#) len) [OLDCALL](#)

20.61.1 Detailed Description

Decompressor for RLE encoded data

Decompresses data which has been compressed with [gbcompress](#) using the `--alg=rle` argument.

20.61.2 Macro Definition Documentation

20.61.2.1 RLE_STOP `#define RLE_STOP 0`

20.61.3 Function Documentation

20.61.3.1 rle_init() `uint8_t rle_init (`
 `void * data)`

Initialize the RLE decompressor with RLE data at address **data**

Parameters

<i>data</i>	Pointer to start of RLE compressed data
-------------	---

See also

[rle_decompress](#)

20.61.3.2 rle_decompress() `uint8_t rle_decompress (`
 `void * dest,`
 `uint8_t len)`

Decompress RLE compressed data into **dest** for length **len** bytes

Parameters

<i>dest</i>	Pointer to destination buffer/address
<i>len</i>	Number of bytes to decompress

Returns

Returns 0 if compression is complete, 1 if there is more data to decompress

Before calling this function [rle_init](#) must be called one time to initialize the RLE decompressor.

Decompresses data which has been compressed with [gbcompress](#) using the `--alg=rle` argument.

See also

[rle_init](#)

20.62 gbdk-lib/include/gbdk/version.h File Reference

Macros

- `#define __GBDK_VERSION 410`

20.62.1 Macro Definition Documentation

20.62.1.1 `__GBDK_VERSION` `#define __GBDK_VERSION 410`

20.63 gbdk-lib/include/limits.h File Reference

Macros

- `#define CHAR_BIT 8 /* bits in a char */`
- `#define SCHAR_MAX 127`
- `#define SCHAR_MIN -128`
- `#define UCHAR_MAX 0xff`
- `#define CHAR_MAX SCHAR_MAX`
- `#define CHAR_MIN SCHAR_MIN`
- `#define INT_MIN (-32767 - 1)`
- `#define INT_MAX 32767`
- `#define SHRT_MAX INT_MAX`
- `#define SHRT_MIN INT_MIN`
- `#define UINT_MAX 0xffff`
- `#define UINT_MIN 0`
- `#define USHRT_MAX UINT_MAX`
- `#define USHRT_MIN UINT_MIN`
- `#define LONG_MIN (-2147483647L-1)`
- `#define LONG_MAX 2147483647L`
- `#define ULONG_MAX 0xffffffff`
- `#define ULONG_MIN 0`

20.63.1 Macro Definition Documentation

20.63.1.1 `CHAR_BIT` `#define CHAR_BIT 8 /* bits in a char */`

20.63.1.2 `SCHAR_MAX` `#define SCHAR_MAX 127`

20.63.1.3 `SCHAR_MIN` `#define SCHAR_MIN -128`

20.63.1.4 `UCHAR_MAX` `#define UCHAR_MAX 0xff`

20.63.1.5 `CHAR_MAX` `#define CHAR_MAX SCHAR_MAX`

20.63.1.6 `CHAR_MIN` `#define CHAR_MIN SCHAR_MIN`

20.63.1.7 `INT_MIN` `#define INT_MIN (-32767 - 1)`

20.63.1.8 `INT_MAX` `#define INT_MAX 32767`

20.63.1.9 SHRT_MAX `#define SHRT_MAX INT_MAX`

20.63.1.10 SHRT_MIN `#define SHRT_MIN INT_MIN`

20.63.1.11 UINT_MAX `#define UINT_MAX 0xffff`

20.63.1.12 UINT_MIN `#define UINT_MIN 0`

20.63.1.13 USHRT_MAX `#define USHRT_MAX UINT_MAX`

20.63.1.14 USHRT_MIN `#define USHRT_MIN UINT_MIN`

20.63.1.15 LONG_MIN `#define LONG_MIN (-2147483647L-1)`

20.63.1.16 LONG_MAX `#define LONG_MAX 2147483647L`

20.63.1.17 ULONG_MAX `#define ULONG_MAX 0xffffffff`

20.63.1.18 ULONG_MIN `#define ULONG_MIN 0`

20.64 gbdk-lib/include/msx/msx.h File Reference

```
#include <types.h>
#include <stdint.h>
#include <gbdk/version.h>
#include <msx/hardware.h>
```

Data Structures

- struct [joypads_t](#)
- struct [OAM_item_t](#)

Macros

- `#define MSX`
- `#define VBK_REG VDP_ATTR_SHIFT`
- `#define J_UP 0b00100000`
- `#define J_DOWN 0b01000000`
- `#define J_LEFT 0b00010000`
- `#define J_RIGHT 0b10000000`
- `#define J_A 0b00000001`
- `#define J_B 0b00000100`
- `#define J_SELECT 0b00001000`
- `#define J_START 0b00000010`
- `#define M_TEXT_OUT 0x02U`

```

• #define M_TEXT_INOUT 0x03U
• #define M_NO_SCROLL 0x04U
• #define M_NO_INTERP 0x08U
• #define S_FLIPX 0x02U
• #define S_FLIPY 0x04U
• #define S_PALETTE 0x08U
• #define S_PRIORITY 0x10U
• #define __WRITE_VDP_REG(REG, v) shadow_##REG=(v);__critical{VDP_CMD=(shadow_##REG),VDP←
  _CMD=REG;}
• #define __READ_VDP_REG(REG) shadow_##REG
• #define EMPTY_IFLAG 0x00U
• #define VBL_IFLAG 0x01U
• #define LCD_IFLAG 0x02U
• #define TIM_IFLAG 0x04U
• #define SIO_IFLAG 0x08U
• #define JOY_IFLAG 0x10U
• #define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH
• #define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT
• #define MINWNDPOSX 0x00U
• #define MINWNDPOSY 0x00U
• #define MAXWNDPOSX 0x00U
• #define MAXWNDPOSY 0x00U
• #define DISPLAY_ON __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) |= R1_DISP_ON)
• #define DISPLAY_OFF display_off();
• #define HIDE_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) |= R0_LCB)
• #define SHOW_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) &=
  (~R0_LCB))
• #define SHOW_BKG
• #define HIDE_BKG
• #define SHOW_WIN
• #define HIDE_WIN
• #define SHOW_SPRITES
• #define HIDE_SPRITES
• #define SPRITES_16x16 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) |= R1_SPR_16X16)
• #define SPRITES_8x8 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) &= (~R1_SPR_16X16))
• #define DEVICE_SUPPORTS_COLOR (TRUE)
• #define CURRENT_BANK _current_bank
• #define BANK(VARNAME) ( (uint8_t) & __bank_ ## VARNAME )
• #define BANKREF(VARNAME)
• #define BANKREF_EXTERN(VARNAME) extern const void __bank_ ## VARNAME;
• #define SWITCH_ROM1 SWITCH_ROM
• #define SWITCH_ROM2(b) MAP_FRAME2=(b)
• #define SWITCH_RAM(b) RAM_CONTROL=((b)&1)?RAM_CONTROL|RAMCTL_BANK:RAM_CONTR←
  OL&(~RAMCTL_BANK)
• #define ENABLE_RAM RAM_CONTROL|=RAMCTL_RAM
• #define DISABLE_RAM RAM_CONTROL&=(~RAMCTL_RAM)
• #define set_bkg_palette_entry set_palette_entry
• #define set_sprite_palette_entry(palette, entry, rgb_data) set_palette_entry(1,entry,rgb_data)
• #define set_bkg_palette set_palette
• #define set_sprite_palette(first_palette, nb_palettes, rgb_data) set_palette(1,1,rgb_data)
• #define COMPAT_PALETTE(C0, C1, C2, C3) (((uint16_t)(C3) << 12) | ((uint16_t)(C2) << 8) |
  ((uint16_t)(C1) << 4) | (uint16_t)(C0))
• #define set_bkg_tiles set_tile_map
• #define set_win_tiles set_tile_map
• #define fill_bkg_rect fill_rect

```

- `#define fill_win_rect fill_rect`
- `#define DISABLE_VBL_TRANSFER _shadow_OAM_base = 0`
- `#define ENABLE_VBL_TRANSFER _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM >> 8)`
- `#define MAX_HARDWARE_SPRITES 32`
- `#define set_bkg_tile_xy set_tile_xy`
- `#define set_win_tile_xy set_tile_xy`
- `#define get_win_xy_addr get_bkg_xy_addr`

Typedefs

- `typedef void(* int_handler) (void) NONBANKED`
- `typedef struct OAM_item_t OAM_item_t`

Functions

- `void WRITE_VDP_CMD (uint16_t cmd) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void WRITE_VDP_DATA (uint16_t data) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void mode (uint8_t m) OLDCALL`
- `uint8_t get_mode () OLDCALL`
- `void set_interrupts (uint8_t flags) Z88DK_FASTCALL`
- `void remove_VBL (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(iyh`
- `void remove_LCD (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void remove_TIM (int_handler h) Z88DK_FASTCALL`
- `void remove_SIO (int_handler h) Z88DK_FASTCALL`
- `void remove_JOY (int_handler h) Z88DK_FASTCALL`
- `void add_VBL (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(d`
- `void add_LCD (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void add_TIM (int_handler h) Z88DK_FASTCALL`
- `void add_SIO (int_handler h) Z88DK_FASTCALL`
- `void add_JOY (int_handler h) Z88DK_FASTCALL`
- `uint8_t cancel_pending_interrupts ()`
- `void move_bkg (uint8_t x, uint8_t y)`
- `void scroll_bkg (int8_t x, int8_t y)`
- `void wait_vbl_done () PRESERVES_REGS(b`
- `void display_off ()`
- `void refresh_OAM ()`
- `void SWITCH_ROM (uint8_t bank) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void delay (uint16_t d) Z88DK_FASTCALL`
- `uint8_t joypad () OLDCALL PRESERVES_REGS(b`
- `uint8_t waitpad (uint8_t mask) Z88DK_FASTCALL PRESERVES_REGS(b`
- `void waitpadup () PRESERVES_REGS(b`
- `uint8_t joypad_init (uint8_t npads, joypads_t *joypads) Z88DK_CALLEE`
- `void joypad_ex (joypads_t *joypads) Z88DK_FASTCALL PRESERVES_REGS(iyh`
- `void set_default_palette ()`
- `void cpu_fast ()`
- `void set_palette_entry (uint8_t palette, uint8_t entry, uint16_t rgb_data) Z88DK_CALLEE PRESERVES_REGS(iyh`
- `void set_palette (uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) Z88DK_CALLEE`
- `void set_native_tile_data (uint16_t start, uint16_t ntiles, const void *src) Z88DK_CALLEE`
- `void set_bkg_4bpp_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_sprite_1bpp_data (uint16_t start, uint16_t ntiles, const void *src) Z88DK_CALLEE`
- `void set_native_sprite_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_2bpp_palette (uint16_t palette)`
- `void set_bkg_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_sprite_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_1bpp_colors (uint8_t fgcolor, uint8_t bgcolor)`

- void `set_tile_1bpp_data` (uint16_t start, uint16_t ntiles, const void *src, uint16_t colors) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_1bpp_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_data` (uint16_t dst, const void *src, uint16_t size) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `vmemcpy` (uint16_t dst, const void *src, uint16_t size) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_tile_map` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_based_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void `set_win_based_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void `set_tile_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t map_w, const uint8_t *map) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_tile_submap_compat` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t map_w, const uint8_t *map) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w)
- void `set_win_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w)
- void `set_bkg_based_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void `set_win_based_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void `fill_rect` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint16_t tile) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `SET_SHADOW_OAM_ADDRESS` (void *address)
- void `set_sprite_tile` (uint8_t nb, uint8_t tile)
- uint8_t `get_sprite_tile` (uint8_t nb)
- void `set_sprite_prop` (uint8_t nb, uint8_t prop)
- uint8_t `get_sprite_prop` (uint8_t nb)
- void `move_sprite` (uint8_t nb, uint8_t x, uint8_t y)
- void `scroll_sprite` (uint8_t nb, int8_t x, int8_t y)
- void `hide_sprite` (uint8_t nb)
- void `set_vram_byte` (uint8_t *addr, uint8_t v) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `set_attributed_tile_xy` (uint8_t x, uint8_t y, uint16_t t) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `set_tile_xy` (uint8_t x, uint8_t y, uint8_t t) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `get_bkg_xy_addr` (uint8_t x, uint8_t y) Z88DK_CALLEE PRESERVES_REGS(iyh)

Variables

- void `c`
- void `d`
- void `e`
- void `iyh`
- void `iyi`
- void `h`
- void `l`
- volatile uint16_t `sys_time`
- volatile uint8_t `current_bank`
- uint16_t `current_2bpp_palette`
- uint16_t `current_1bpp_colors`
- uint8_t `map_tile_offset`
- uint8_t `submap_tile_offset`
- volatile struct `OAM_item_t shadow_OAM` []
- volatile uint8_t `shadow_OAM_base`
- volatile uint8_t `shadow_OAM_OFF`

20.64.1 Detailed Description

MSX specific functions.

20.64.2 Macro Definition Documentation

20.64.2.1 MSX `#define MSX`

20.64.2.2 VBK_REG `#define VBK_REG VDP_ATTR_SHIFT`

20.64.2.3 J_UP `#define J_UP 0b00100000`

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_t keys; keys = joypad(); if (keys & J_B) { ... }
```

See also

[joypad](#)

20.64.2.4 J_DOWN `#define J_DOWN 0b01000000`

20.64.2.5 J_LEFT `#define J_LEFT 0b00010000`

20.64.2.6 J_RIGHT `#define J_RIGHT 0b10000000`

20.64.2.7 J_A `#define J_A 0b00000001`

20.64.2.8 J_B `#define J_B 0b00000100`

20.64.2.9 J_SELECT `#define J_SELECT 0b00001000`

20.64.2.10 J_START `#define J_START 0b00000010`

20.64.2.11 M_TEXT_OUT `#define M_TEXT_OUT 0x02U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

20.64.2.12 M_TEXT_INOUT `#define M_TEXT_INOUT 0x03U`

20.64.2.13 M_NO_SCROLL `#define M_NO_SCROLL 0x04U`

Set this in addition to the others to disable scrolling

If scrolling is disabled, the cursor returns to (0,0)

See also

[mode\(\)](#)

20.64.2.14 M_NO_INTERP `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

20.64.2.15 S_FLIPX `#define S_FLIPX 0x02U`

If set the background tile will be flipped horizontally.

20.64.2.16 S_FLIPY `#define S_FLIPY 0x04U`

If set the background tile will be flipped vertically.

20.64.2.17 S_PALETTE `#define S_PALETTE 0x08U`

If set the background tile palette.

20.64.2.18 S_PRIORITY `#define S_PRIORITY 0x10U`

If set the background tile priority.

20.64.2.19 __WRITE_VDP_REG `#define __WRITE_VDP_REG(`

REG,

v) shadow_##REG=(v);__critical{VDP_CMD=(shadow_##REG),VDP_CMD=REG;}

20.64.2.20 __READ_VDP_REG `#define __READ_VDP_REG(`

REG) shadow_##REG

20.64.2.21 EMPTY_IFLAG `#define EMPTY_IFLAG 0x00U`

Disable calling of interrupt service routines

20.64.2.22 VBL_IFLAG `#define VBL_IFLAG 0x01U`

VBlank Interrupt occurs at the start of the vertical blank.

During this period the video ram may be freely accessed.

See also

[set_interrupts\(\)](#),

[add_VBL](#)

20.64.2.23 LCD_IFLAG `#define LCD_IFLAG 0x02U`

LCD Interrupt when triggered by the STAT register.

See also

[set_interrupts\(\)](#),

[add_LCD](#)

20.64.2.24 TIM_IFLAG `#define TIM_IFLAG 0x04U`
Does nothing on MSX

20.64.2.25 SIO_IFLAG `#define SIO_IFLAG 0x08U`
Does nothing on MSX

20.64.2.26 JOY_IFLAG `#define JOY_IFLAG 0x10U`
Does nothing on MSX

20.64.2.27 SCREENWIDTH `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`
Width of the visible screen in pixels.

20.64.2.28 SCREENHEIGHT `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`
Height of the visible screen in pixels.

20.64.2.29 MINWNDPOSX `#define MINWNDPOSX 0x00U`
The Minimum X position of the Window Layer (Left edge of screen)

See also

[move_win\(\)](#)

20.64.2.30 MINWNDPOSY `#define MINWNDPOSY 0x00U`
The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move_win\(\)](#)

20.64.2.31 MAXWNDPOSX `#define MAXWNDPOSX 0x00U`
The Maximum X position of the Window Layer (Right edge of screen)

See also

[move_win\(\)](#)

20.64.2.32 MAXWNDPOSY `#define MAXWNDPOSY 0x00U`
The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move_win\(\)](#)

20.64.2.33 DISPLAY_ON `#define DISPLAY_ON __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) |= R1_DISP_ON)`
Turns the display back on.

See also

[display_off](#), [DISPLAY_OFF](#)

20.64.2.34 DISPLAY_OFF `#define DISPLAY_OFF display_off();`
Turns the display off immediately.

See also

[display_off](#), [DISPLAY_ON](#)

20.64.2.35 HIDE_LEFT_COLUMN `#define HIDE_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) | R0_LCB)`

Blanks leftmost column, so it is not garbaged when you use horizontal scroll

See also

[SHOW_LEFT_COLUMN](#)

20.64.2.36 SHOW_LEFT_COLUMN `#define SHOW_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) &= (~R0_LCB))`

Shows leftmost column

See also

[HIDE_LEFT_COLUMN](#)

20.64.2.37 SHOW_BKG `#define SHOW_BKG`
Turns on the background layer. Not yet implemented

20.64.2.38 HIDE_BKG `#define HIDE_BKG`
Turns off the background layer. Not yet implemented

20.64.2.39 SHOW_WIN `#define SHOW_WIN`
Turns on the window layer Not yet implemented

20.64.2.40 HIDE_WIN `#define HIDE_WIN`
Turns off the window layer. Not yet implemented

20.64.2.41 SHOW_SPRITES `#define SHOW_SPRITES`
Turns on the sprites layer. Not yet implemented

20.64.2.42 HIDE_SPRITES `#define HIDE_SPRITES`
Turns off the sprites layer. Not yet implemented

20.64.2.43 SPRITES_16x16 `#define SPRITES_16x16 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) | R1_SPR_16X16)`
Sets sprite size to 8x16 pixels, two tiles one above the other.

20.64.2.44 SPRITES_8x8 `#define SPRITES_8x8 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) &= (~R1_SPR_16X16))`
Sets sprite size to 8x8 pixels, one tile.

20.64.2.45 DEVICE_SUPPORTS_COLOR `#define DEVICE_SUPPORTS_COLOR (TRUE)`
Macro returns TRUE if device supports color (it always does on MSX)

20.64.2.46 CURRENT_BANK `#define CURRENT_BANK _current_bank`

20.64.2.47 BANK `#define BANK(`
`VARNAME) ((uint8_t) & __bank_ ## VARNAME)`

Obtains the **bank number** of VARNAME

Parameters

<i>VARNAME</i>	Name of the variable which has a <code>__bank_VARNAME</code> companion symbol which is adjusted by bankpack
----------------	---

Use this to obtain the bank number from a bank reference created with [BANKREF\(\)](#).

See also

[BANKREF_EXTERN\(\)](#), [BANKREF\(\)](#)

20.64.2.48 BANKREF `#define BANKREF(`
`VARNAME)`

Value:

```
void __func_ ## VARNAME() __banked __naked { \
__asm \
    .local b__func_ ## VARNAME \
    __bank_ ## VARNAME = b__func_ ## VARNAME \
    .globl __bank_ ## VARNAME \
__endasm; \
}
```

Creates a reference for retrieving the bank number of a variable or function

Parameters

<i>VARNAME</i>	Variable name to use, which may be an existing identifier
----------------	---

See also

[BANK\(\)](#) for obtaining the bank number of the included data.

More than one [BANKREF \(\)](#) may be created per file, but each call should always use a unique VARNAME. Use [BANKREF_EXTERN\(\)](#) within another source file to make the variable and it's data accesible there.

20.64.2.49 BANKREF_EXTERN `#define BANKREF_EXTERN(`
`VARNAME) extern const void __bank_ ## VARNAME;`

Creates extern references for accessing a [BANKREF\(\)](#) generated variable.

Parameters

<i>VARNAME</i>	Name of the variable used with BANKREF()
----------------	--

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

20.64.2.50 SWITCH_ROM1 `#define SWITCH_ROM1 SWITCH_ROM`

20.64.2.51 SWITCH_ROM2 `#define SWITCH_ROM2(
 b) MAP_FRAME2=(b)`

Makes switch the active ROM bank in frame 2

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

20.64.2.52 SWITCH_RAM `#define SWITCH_RAM(
 b) RAM_CONTROL=((b) & 1) ? RAM_CONTROL | RAMCTL_BANK : RAM_CONTROL & (~RAMCTL_BANK)`

Switches RAM bank

Parameters

<i>b</i>	SRAM bank to switch to
----------	------------------------

20.64.2.53 ENABLE_RAM `#define ENABLE_RAM RAM_CONTROL|=RAMCTL_RAM`
Enables RAM

20.64.2.54 DISABLE_RAM `#define DISABLE_RAM RAM_CONTROL&=(~RAMCTL_RAM)`
Disables RAM

20.64.2.55 set_bkg_palette_entry `#define set_bkg_palette_entry set_palette_entry`

20.64.2.56 set_sprite_palette_entry `#define set_sprite_palette_entry(
 palette,
 entry,
 rgb_data) set_palette_entry(1,entry,rgb_data)`

20.64.2.57 set_bkg_palette `#define set_bkg_palette set_palette`

20.64.2.58 set_sprite_palette `#define set_sprite_palette(
 first_palette,
 nb_palettes,
 rgb_data) set_palette(1,1,rgb_data)`

20.64.2.59 COMPAT_PALETTE `#define COMPAT_PALETTE(
 C0,
 C1,
 C2,
 C3) (((uint16_t) (C3) << 12) | ((uint16_t) (C2) << 8) | ((uint16_t) (C1) << 4) |
 (uint16_t) (C0))`

20.64.2.60 set_bkg_tiles `#define set_bkg_tiles set_tile_map`

20.64.2.61 set_win_tiles `#define set_win_tiles set_tile_map`

20.64.2.62 fill_bkg_rect `#define fill_bkg_rect fill_rect`

20.64.2.63 fill_win_rect `#define fill_win_rect fill_rect`

20.64.2.64 DISABLE_VBL_TRANSFER `#define DISABLE_VBL_TRANSFER _shadow_OAM_base = 0`
Disable shadow OAM to VRAM copy on each VBlank

20.64.2.65 ENABLE_VBL_TRANSFER `#define ENABLE_VBL_TRANSFER _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM_base >> 8)`
Enable shadow OAM to VRAM copy on each VBlank

20.64.2.66 MAX_HARDWARE_SPRITES `#define MAX_HARDWARE_SPRITES 32`
Amount of hardware sprites in OAM

20.64.2.67 set_bkg_tile_xy `#define set_bkg_tile_xy set_tile_xy`

20.64.2.68 set_win_tile_xy `#define set_win_tile_xy set_tile_xy`

20.64.2.69 get_win_xy_addr `#define get_win_xy_addr get_bkg_xy_addr`

20.64.3 Typedef Documentation

20.64.3.1 int_handler `typedef void(* int_handler) (void) NONBANKED`
Interrupt handlers

20.64.3.2 OAM_item_t `typedef struct OAM_item_t OAM_item_t`
Sprite Attributes structure

Parameters

<i>x</i>	X Coordinate of the sprite on screen
<i>y</i>	Y Coordinate of the sprite on screen
<i>tile</i>	Sprite tile number (see set_sprite_tile)
<i>prop</i>	OAM Property Flags (see set_sprite_prop)

20.64.4 Function Documentation

20.64.4.1 WRITE_VDP_CMD() `void WRITE_VDP_CMD (uint16_t cmd)`

20.64.4.2 WRITE_VDP_DATA() `void WRITE_VDP_DATA (uint16_t data)`

20.64.4.3 mode() `void mode (`
`uint8_t m)`

Set the current screen mode - one of M_* modes
Normally used by internal functions only.

See also

[M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.64.4.4 get_mode() `uint8_t get_mode ()`

Returns the current mode

See also

[M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

Returns the current mode

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.64.4.5 set_interrupts() `void set_interrupts (`
`uint8_t flags)`

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

Parameters

<i>flags</i>	A logical OR of *_IFLAGS
--------------	--------------------------

Note

: This disables and then re-enables interrupts so it must be used outside of a critical section.

See also

[enable_interrupts\(\)](#), [disable_interrupts\(\)](#)
[VBL_IFLAG](#), [LCD_IFLAG](#), [TIM_IFLAG](#), [SIO_IFLAG](#), [JOY_IFLAG](#)

20.64.4.6 remove_VBL() `void remove_VBL (`
`int_handler h)`

Removes the VBL interrupt handler.

See also

[add_VBL\(\)](#)

20.64.4.7 remove_LCD() `void remove_LCD (`
`int_handler h)`

Removes the LCD interrupt handler.

See also

[add_LCD\(\)](#), [remove_VBL\(\)](#)

20.64.4.8 remove_TIM() void remove_TIM (
int_handler h)

20.64.4.9 remove_SIO() void remove_SIO (
int_handler h)

20.64.4.10 remove_JOY() void remove_JOY (
int_handler h)

20.64.4.11 add_VBL() void add_VBL (
int_handler h)

Adds a V-blank interrupt handler.

20.64.4.12 add_LCD() void add_LCD (
int_handler h)

Adds a LCD interrupt handler.

20.64.4.13 add_TIM() void add_TIM (
int_handler h)

Does nothing on MSX

20.64.4.14 add_SIO() void add_SIO (
int_handler h)

Does nothing on MSX

20.64.4.15 add_JOY() void add_JOY (
int_handler h)

Does nothing on MSX

20.64.4.16 cancel_pending_interrupts() uint8_t cancel_pending_interrupts () [inline]
Cancel pending interrupts

20.64.4.17 move_bkg() void move_bkg (
uint8_t x,
uint8_t y) [inline]

20.64.4.18 scroll_bkg() void scroll_bkg (
int8_t x,
int8_t y) [inline]

20.64.4.19 wait_vbl_done() void wait_vbl_done ()

HALTs the CPU and waits for the vertical blank interrupt (VBL) to finish.

This is often used in main loops to idle the CPU at low power until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediately.

20.64.4.20 display_off() `void display_off () [inline]`
Turns the display off.

See also

[DISPLAY_ON](#)

20.64.4.21 refresh_OAM() `void refresh_OAM ()`
Copies data from shadow OAM to OAM

20.64.4.22 SWITCH_ROM() `void SWITCH_ROM (`
 `uint8_t bank)`
Makes switch the active ROM bank in frame 1

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

20.64.4.23 delay() `void delay (`
 `uint16_t d)`
Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

20.64.4.24 joypad() `uint8_t joypad ()`
Reads and returns the current state of the joypad.

20.64.4.25 waitpad() `uint8_t waitpad (`
 `uint8_t mask)`
Waits until at least one of the buttons given in mask are pressed.

20.64.4.26 waitpadup() `void waitpadup ()`
Waits for the directional pad and all buttons to be released.
Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

20.64.4.27 joypad_init() `uint8_t joypad_init (`
 `uint8_t npads,`
 `joypads_t * joypads)`
Initializes [joypads_t](#) structure for polling multiple joypads

Parameters

<i>npads</i>	number of joypads requested (1, 2 or 4)
<i>joypads</i>	pointer to joypads_t structure to be initialized

Only required for [joypad_ex](#), not required for calls to regular [joypad\(\)](#)

Returns

number of joypads available

See also

[joypad_ex\(\)](#), [joypads_t](#)

20.64.4.28 joypad_ex() `void joypad_ex (`
 `joypads_t * joypads)`

Polls all available joypads

Parameters

<code>joypads</code>	pointer to <code>joypads_t</code> structure to be filled with joypad statuses, must be previously initialized with <code>joypad_init()</code>
----------------------	---

See also

`joypad_init()`, `joypads_t`

20.64.4.29 set_default_palette() `void set_default_palette ()`

20.64.4.30 cpu_fast() `void cpu_fast () [inline]`

Set CPU speed to fast (CGB Double Speed) operation.

On startup the CGB operates in Normal Speed Mode and can be switched into Double speed mode (faster processing but also higher power consumption). See the Pan Docs for more information about which hardware features operate faster and which remain at Normal Speed.

- Interrupts are temporarily disabled and then re-enabled during this call.
- You can check to see if `_cpu == CGB_TYPE` before using this function.

See also

`cpu_slow()`, `_cpu`

20.64.4.31 set_palette_entry() `void set_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `uint16_t rgb_data)`

20.64.4.32 set_palette() `void set_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

20.64.4.33 set_native_tile_data() `void set_native_tile_data (`
 `uint16_t start,`
 `uint16_t ntiles,`
 `const void * src)`

20.64.4.34 set_bkg_4bpp_data() `void set_bkg_4bpp_data (`
 `uint16_t start,`
 `uint16_t ntiles,`
 `const void * src) [inline]`

20.64.4.35 set_sprite_1bpp_data() void set_sprite_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.64.4.36 set_native_sprite_data() void set_native_sprite_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.64.4.37 set_2bpp_palette() void set_2bpp_palette (
 uint16_t palette) [inline]

20.64.4.38 set_bkg_data() void set_bkg_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.64.4.39 set_sprite_data() void set_sprite_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.64.4.40 set_1bpp_colors() void set_1bpp_colors (
 uint8_t fgcolor,
 uint8_t bgcolor) [inline]

20.64.4.41 set_tile_1bpp_data() void set_tile_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src,
 uint16_t colors)

20.64.4.42 set_bkg_1bpp_data() void set_bkg_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.64.4.43 set_data() void set_data (
 uint16_t dst,
 const void * src,
 uint16_t size)

Copies arbitrary data to an address in VRAM

Parameters

<i>dst</i>	destination VRAM Address
<i>src</i>	Pointer to source buffer
<i>size</i>	Number of bytes to copy

Copies **size** bytes from a buffer at **_src__** to VRAM starting at **dst**.

20.64.4.44 vmemcpy() `void vmemcpy (`
 `uint16_t dst,`
 `const void * src,`
 `uint16_t size)`

20.64.4.45 set_tile_map() `void set_tile_map (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `const uint8_t * tiles)`

20.64.4.46 set_bkg_based_tiles() `void set_bkg_based_tiles (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `const uint8_t * tiles,`
 `uint8_t base_tile) [inline]`

20.64.4.47 set_win_based_tiles() `void set_win_based_tiles (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `const uint8_t * tiles,`
 `uint8_t base_tile) [inline]`

20.64.4.48 set_tile_submap() `void set_tile_submap (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t map_w,`
 `const uint8_t * map)`

20.64.4.49 set_tile_submap_compat() `void set_tile_submap_compat (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t map_w,`
 `const uint8_t * map)`

20.64.4.50 set_bkg_submap() `void set_bkg_submap (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`

```
uint8_t h,
const uint8_t * map,
uint8_t map_w ) [inline]
```

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 32 tiles.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: $x \& 0x1F$ and $y \& 0x1F$). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: $(map_ptr + x + (y * map_width))$.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: $map_ptr + 1 + (2 * map_width)$

Use this instead of [set_bkg_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See [set_bkg_tiles](#) for setting CGB attribute maps with [VBK_REG](#).

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_tiles](#), [set_win_submap](#), [set_tiles](#)

20.64.4.51 set_win_submap()

```
void set_win_submap (
uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * map,
uint8_t map_w ) [inline]
```

Sets a rectangular area of the Window Tile Map using a sub-region from a source tile map.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Window Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: $x \& 0x1F$ and $y \& 0x1F$). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: $(map_ptr + x + (y * map_width))$.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: $map_ptr + 1 + (2 * map_width)$

Use this instead of [set_win_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- [VBK_REG](#) = [VBK_TILES](#) Tile Numbers are written
- [VBK_REG](#) = [VBK_ATTRIBUTES](#) Tile Attributes are written

See [set_bkg_tiles](#) for details about CGB attribute maps with [VBK_REG](#).

See also

[SHOW_WIN](#), [HIDE_WIN](#), [set_win_tiles](#), [set_bkg_submap](#), [set_bkg_tiles](#), [set_bkg_data](#), [set_tiles](#)

20.64.4.52 set_bkg_based_submap() `void set_bkg_based_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w,`
`uint8_t base_tile) [inline]`

20.64.4.53 set_win_based_submap() `void set_win_based_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w,`
`uint8_t base_tile) [inline]`

20.64.4.54 fill_rect() `void fill_rect (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint16_t tile)`

20.64.4.55 SET_SHADOW_OAM_ADDRESS() `void SET_SHADOW_OAM_ADDRESS (`
`void * address) [inline]`

Sets address of 256-byte aligned array of shadow OAM to be transferred on each VBlank

20.64.4.56 set_sprite_tile() `void set_sprite_tile (`
`uint8_t nb,`
`uint8_t tile) [inline]`

Sets sprite number **nb** in the OAM to display tile number **tile**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>tile</i>	Selects a tile (0 - 255) from memory at 8000h - 8FFFh In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the OAM Attribute Flag (see set_sprite_prop)

In 8x16 mode:

- The sprite will also display the next tile (**tile** + 1) directly below (y + 8) the first tile.
- The lower bit of the tile number is ignored: the upper 8x8 tile is (**tile** & 0xFE), and the lower 8x8 tile is (**tile** | 0x01).
- See: [SPRITES_8x16](#)

20.64.4.57 get_sprite_tile() `uint8_t get_sprite_tile (`
`uint8_t nb) [inline]`

Returns the tile number of sprite number **nb** in the OAM.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[set_sprite_tile](#) for more details

20.64.4.58 set_sprite_prop() `void set_sprite_prop (`
`uint8_t nb,`
`uint8_t prop) [inline]`

20.64.4.59 get_sprite_prop() `uint8_t get_sprite_prop (`
`uint8_t nb) [inline]`

20.64.4.60 move_sprite() `void move_sprite (`
`uint8_t nb,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves sprite number **nb** to the **x, y** position on the screen.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	X Position. Specifies the sprites horizontal position on the screen (minus 8). An offscreen value (X=0 or X>=168) hides the sprite, but the sprite still affects the priority ordering - a better way to hide a sprite is to set its Y-coordinate offscreen.

Parameters

<i>y</i>	Y Position. Specifies the sprites vertical position on the screen (minus 16). An offscreen value (for example, Y=0 or Y>=160) hides the sprite.
----------	--

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

20.64.4.61 scroll_sprite() `void scroll_sprite (`
 `uint8_t nb,`
 `int8_t x,`
 `int8_t y) [inline]`

Moves sprite number **nb** relative to its current position.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	Number of pixels to move the sprite on the X axis Range: -128 - 127
<i>y</i>	Number of pixels to move the sprite on the Y axis Range: -128 - 127

See also

[move_sprite](#) for more details about the X and Y position

20.64.4.62 hide_sprite() `void hide_sprite (`
 `uint8_t nb) [inline]`

Hides sprite number **nb** by moving it to zero position by Y.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

20.64.4.63 set_vram_byte() `void set_vram_byte (`
 `uint8_t * addr,`
 `uint8_t v)`

Set byte in vram at given memory location

Parameters

<i>addr</i>	address to write to
<i>v</i>	value

20.64.4.64 set_attributed_tile_xy() `uint8_t* set_attributed_tile_xy (`
 `uint8_t x,`
 `uint8_t y,`
 `uint16_t t)`

Set single tile t with attributes on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

```
20.64.4.65  set_tile_xy()  uint8_t* set_tile_xy (
    uint8_t x,
    uint8_t y,
    uint8_t t )
```

Set single tile t on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

```
20.64.4.66  get_bkg_xy_addr()  uint8_t* get_bkg_xy_addr (
    uint8_t x,
    uint8_t y )
```

Get address of X,Y tile of background map

20.64.5 Variable Documentation

20.64.5.1 **c** void c

20.64.5.2 **d** void d

20.64.5.3 **e** void e

20.64.5.4 **iyh** void iyh

20.64.5.5 **iy1** uint8_t iy1

20.64.5.6 **h** uint8_t h

20.64.5.7 | `void l`**20.64.5.8** `sys_time` `volatile uint16_t sys_time`

Global Time Counter in VBL periods (60Hz)

Increments once per Frame

Will wrap around every ~18 minutes (unsigned 16 bits = 65535 / 60 / 60 = 18.2)

20.64.5.9 `_current_bank` `volatile uint8_t _current_bank`

Tracks current active ROM bank in frame 1

Tracks current active ROM bank

The active bank number is not tracked by `_current_bank` when `SWITCH_ROM_MBC5_8M` is used.

This variable is updated automatically when you call `SWITCH_ROM_MBC1` or `SWITCH_ROM_MBC5`, `SWITCH_ROM()`, or call a BANKED function.

See also

`SWITCH_ROM_MBC1()`, `SWITCH_ROM_MBC5()`, `SWITCH_ROM()`

20.64.5.10 `_current_2bpp_palette` `uint16_t _current_2bpp_palette`**20.64.5.11** `_current_1bpp_colors` `uint16_t _current_1bpp_colors`**20.64.5.12** `_map_tile_offset` `uint8_t _map_tile_offset`**20.64.5.13** `_submap_tile_offset` `uint8_t _submap_tile_offset`**20.64.5.14** `shadow_OAM` `volatile struct OAM_item_t shadow_OAM[]`

Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

20.64.5.15 `_shadow_OAM_base` `volatile uint8_t _shadow_OAM_base`

MSB of shadow_OAM address is used by OAM copying routine

MSB of shadow_OAM address is used by OAM DMA copying routine

20.64.5.16 `_shadow_OAM_OFF` `volatile uint8_t _shadow_OAM_OFF`

Flag for disabling of OAM copying routine

Values:

- 1: OAM copy routine is disabled (non-isr VDP operation may be in progress)
- 0: OAM copy routine is enabled

This flag is modified by all MSX GBDK API calls that write to the VDP. It is set to DISABLED when they start and ENABLED when they complete.

Note

It is recommended to avoid writing to the Video Display Processor (VDP) during an interrupt service routine (ISR) since it can corrupt the VDP pointer of an VDP operation already in progress.

If it is necessary, this flag can be used during an ISR to determine whether a VDP operation is already in progress. If the value is 1 then avoid writing to the VDP (tiles, map, scrolling, colors, etc).

```
// at the beginning of and ISR that would write to the VDP
if (_shadow_OAM_OFF) return;
```

See also

[docs_consoles_safe_display_controller_access](#)

20.65 gbdk-lib/include/nes/nes.h File Reference

```
#include <types.h>
#include <stdint.h>
#include <gbdk/version.h>
#include <nes/hardware.h>
#include <nes/rgb_to_nes_macro.h>
```

Data Structures

- struct [joypads_t](#)
- struct [OAM_item_t](#)

Macros

- #define [NINTENDO_ENTERTAINMENT_SYSTEM](#)
- #define [RGB\(r, g, b\)](#) [RGB_TO_NES](#)((((r) >> 2) | ((g) << 2) | ((b) << 4)))
- #define [RGB8\(r, g, b\)](#) [RGB_TO_NES](#)((((r) >> 6) | (((g) >> 6) << 2) | (((b) >> 6) << 4)))
- #define [RGBHTML](#)(RGB24bit) [RGB_TO_NES](#)(((((RGB24bit) >> 22) | (((RGB24bit) & 0xFFFF) >> 14) << 2) | (((RGB24bit) & 0xFF) >> 6) << 4)))
- #define [RGB_RED](#) 0x16
- #define [RGB_DARKRED](#) 0x06
- #define [RGB_GREEN](#) 0x2A
- #define [RGB_DARKGREEN](#) 0x1A
- #define [RGB_BLUE](#) 0x12
- #define [RGB_DARKBLUE](#) 0x02
- #define [RGB_YELLOW](#) 0x28
- #define [RGB_DARKYELLOW](#) 0x18
- #define [RGB_CYAN](#) 0x2C
- #define [RGB_AQUA](#) 0x1C
- #define [RGB_PINK](#) 0x24
- #define [RGB_PURPLE](#) 0x14
- #define [RGB_BLACK](#) 0x0F
- #define [RGB_DARKGRAY](#) 0x00
- #define [RGB_LIGHTGRAY](#) 0x10
- #define [RGB_WHITE](#) 0x30
- #define [J_UP](#) 0x10U
- #define [J_DOWN](#) 0x20U
- #define [J_LEFT](#) 0x40U
- #define [J_RIGHT](#) 0x80U
- #define [J_A](#) 0x01U
- #define [J_B](#) 0x02U
- #define [J_SELECT](#) 0x04U
- #define [J_START](#) 0x08U
- #define [M_DRAWING](#) 0x01U
- #define [M_TEXT_OUT](#) 0x02U
- #define [M_TEXT_INOUT](#) 0x03U
- #define [M_NO_SCROLL](#) 0x04U
- #define [M_NO_INTERP](#) 0x08U
- #define [S_PALETTE](#) 0x10U
- #define [S_FLIPX](#) 0x40U
- #define [S_FLIPY](#) 0x80U

- `#define S_PRIORITY 0x20U`
- `#define DMG_BLACK 0x03`
- `#define DMG_DARK_GRAY 0x02`
- `#define DMG_LITE_GRAY 0x01`
- `#define DMG_WHITE 0x00`
- `#define DMG_PALETTE(C0, C1, C2, C3) (((uint8_t)((C3 & 0x03) << 6) | ((C2 & 0x03) << 4) | ((C1 & 0x03) << 2) | ((C0 & 0x03)))`
- `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`
- `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`
- `#define CURRENT_BANK _current_bank`
- `#define BANK(VARNAME) 0`
- `#define BANKREF(VARNAME)`
- `#define BANKREF_EXTERN(VARNAME) extern const void __bank_ ## VARNAME;`
- `#define SWITCH_ROM_DUMMY(b)`
- `#define SWITCH_ROM SWITCH_ROM_DUMMY`
- `#define DISPLAY_ON display_on();`
- `#define DISPLAY_OFF display_off();`
- `#define HIDE_LEFT_COLUMN shadow_PPUMASK &= ~(PPUMASK_SHOW_BG_LC | PPUMASK_SHOW_SPR_LC);`
`\`
- `#define SHOW_LEFT_COLUMN shadow_PPUMASK |= (PPUMASK_SHOW_BG_LC | PPUMASK_SHOW_SPR_LC);`
- `#define SHOW_BKG shadow_PPUMASK |= PPUMASK_SHOW_BG;`
- `#define HIDE_BKG shadow_PPUMASK &= ~PPUMASK_SHOW_BG;`
- `#define SHOW_SPRITES shadow_PPUMASK |= PPUMASK_SHOW_SPR;`
- `#define HIDE_SPRITES shadow_PPUMASK &= ~PPUMASK_SHOW_SPR;`
- `#define SPRITES_8x16 shadow_PPUCTRL |= PPUCTRL_SPR_8X16;`
- `#define SPRITES_8x8 shadow_PPUCTRL &= ~PPUCTRL_SPR_8X16;`
- `#define COMPAT_PALETTE(C0, C1, C2, C3) (((uint8_t)((C3 << 6) | ((C2 << 4) | ((C1 << 2) | (C0))))`
- `#define set_bkg_2bpp_data set_bkg_data`
- `#define set_tile_map set_bkg_tiles`
- `#define set_tile_submap set_bkg_submap`
- `#define set_tile_xy set_bkg_tile_xy`
- `#define set_sprite_2bpp_data set_sprite_data`
- `#define DISABLE_OAM_DMA _shadow_OAM_base = 0`
- `#define DISABLE_VBL_TRANSFER DISABLE_OAM_DMA`
- `#define ENABLE_OAM_DMA _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM >> 8)`
- `#define ENABLE_VBL_TRANSFER ENABLE_OAM_DMA`
- `#define MAX_HARDWARE_SPRITES 64`
- `#define fill_rect fill_bkg_rect`

Typedefs

- `typedef uint8_t palette_color_t`
- `typedef struct OAM_item_t OAM_item_t`

Functions

- `void set_bkg_palette(uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) OLDCALL`
- `void set_sprite_palette(uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) OLDCALL`
- `void set_bkg_palette_entry(uint8_t palette, uint8_t entry, palette_color_t rgb_data) OLDCALL`
- `void set_sprite_palette_entry(uint8_t palette, uint8_t entry, palette_color_t rgb_data) OLDCALL`
- `void mode(uint8_t m) OLDCALL`
- `uint8_t get_mode() OLDCALL`
- `void delay(uint16_t d) OLDCALL`
- `uint8_t joypad() OLDCALL`
- `uint8_t waitpad(uint8_t mask) OLDCALL`

- void `waitpadup` ()
- `uint8_t joypad_init` (`uint8_t npads`, `joypads_t *joypads`) `OLDCALL`
- void `joypad_ex` (`joypads_t *joypads`) `OLDCALL`
- void `enable_interrupts` ()
- void `disable_interrupts` ()
- void `wait_vbl_done` ()
- void `display_off` ()
- void `refresh_OAM` ()
- void `set_vram_byte` (`uint8_t *addr`, `uint8_t v`) `OLDCALL`
- `uint8_t * get_bkg_xy_addr` (`uint8_t x`, `uint8_t y`) `OLDCALL`
- void `set_2bpp_palette` (`uint16_t palette`)
- void `set_1bpp_colors_ex` (`uint8_t fgcolor`, `uint8_t bgcolor`, `uint8_t mode`) `OLDCALL`
- void `set_1bpp_colors` (`uint8_t fgcolor`, `uint8_t bgcolor`)
- void `set_bkg_data` (`uint8_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`) `OLDCALL`
- void `set_bkg_1bpp_data` (`uint8_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`) `OLDCALL`
- void `set_bkg_tiles` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `const uint8_t *tiles`) `OLDCALL`
- void `set_bkg_attributes` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `const uint8_t *attributes`) `OLDCALL`
- void `set_bkg_based_tiles` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `const uint8_t *tiles`, `uint8_t base_tile`)
- void `set_bkg_submap` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `const uint8_t *map`, `uint8_t map_w`) `OLDCALL`
- void `set_bkg_based_submap` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `const uint8_t *map`, `uint8_t map_w`, `uint8_t base_tile`)
- void `get_bkg_tiles` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `uint8_t *tiles`) `OLDCALL`
- `uint8_t * set_bkg_tile_xy` (`uint8_t x`, `uint8_t y`, `uint8_t t`) `OLDCALL`
- `uint8_t get_bkg_tile_xy` (`uint8_t x`, `uint8_t y`) `OLDCALL`
- void `move_bkg` (`uint8_t x`, `uint8_t y`)
- void `scroll_bkg` (`int8_t x`, `int8_t y`)
- void `set_sprite_data` (`uint8_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`) `OLDCALL`
- void `set_sprite_1bpp_data` (`uint8_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`) `OLDCALL`
- void `SET_SHADOW_OAM_ADDRESS` (`void *address`)
- void `set_sprite_tile` (`uint8_t nb`, `uint8_t tile`)
- `uint8_t get_sprite_tile` (`uint8_t nb`)
- void `set_sprite_prop` (`uint8_t nb`, `uint8_t prop`)
- `uint8_t get_sprite_prop` (`uint8_t nb`)
- void `move_sprite` (`uint8_t nb`, `uint8_t x`, `uint8_t y`)
- void `scroll_sprite` (`uint8_t nb`, `int8_t x`, `int8_t y`)
- void `hide_sprite` (`uint8_t nb`)
- void `set_data` (`uint8_t *vram_addr`, `const uint8_t *data`, `uint16_t len`) `OLDCALL`
- void `set_tiles` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `uint8_t *vram_addr`, `const uint8_t *tiles`) `OLDCALL`
- void `set_tile_data` (`uint8_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`, `uint8_t base`) `OLDCALL`
- void `set_native_tile_data` (`uint16_t first_tile`, `uint8_t nb_tiles`, `const uint8_t *data`)
- void `init_bkg` (`uint8_t c`) `OLDCALL`
- void `vmemset` (`void *s`, `uint8_t c`, `size_t n`) `OLDCALL`
- void `fill_bkg_rect` (`uint8_t x`, `uint8_t y`, `uint8_t w`, `uint8_t h`, `uint8_t tile`) `OLDCALL`

Variables

- volatile `uint16_t sys_time`
- volatile `uint8_t _current_bank`
- `uint16_t _current_1bpp_colors`
- `uint8_t _map_tile_offset`
- `uint8_t _submap_tile_offset`
- volatile struct `OAM_item_t shadow_OAM` []
- `uint8_t _shadow_OAM_base`

20.65.1 Detailed Description

NES specific functions.

20.65.2 Macro Definition Documentation

20.65.2.1 NINTENDO_ENTERTAINMENT_SYSTEM `#define NINTENDO_ENTERTAINMENT_SYSTEM`

20.65.2.2 RGB `#define RGB(
 r,
 g,
 b) RGB_TO_NES(((r) | ((g) << 2) | ((b) << 4)))`

20.65.2.3 RGB8 `#define RGB8(
 r,
 g,
 b) RGB_TO_NES((((r) >> 6) | ((g) >> 6) << 2) | (((b) >> 6) << 4)))`

20.65.2.4 RGBHTML `#define RGBHTML(
 RGB24bit) RGB_TO_NES((((RGB24bit) >> 22) | (((RGB24bit) & 0xFFFF) >> 14) <<
2) | (((RGB24bit) & 0xFF) >> 6) << 4)))`

20.65.2.5 RGB_RED `#define RGB_RED 0x16`
Common colors based on the EGA default palette.
Manually entered from https://www.nesdev.org/wiki/PPU_palettes#RGBI

20.65.2.6 RGB_DARKRED `#define RGB_DARKRED 0x06`

20.65.2.7 RGB_GREEN `#define RGB_GREEN 0x2A`

20.65.2.8 RGB_DARKGREEN `#define RGB_DARKGREEN 0x1A`

20.65.2.9 RGB_BLUE `#define RGB_BLUE 0x12`

20.65.2.10 RGB_DARKBLUE `#define RGB_DARKBLUE 0x02`

20.65.2.11 RGB_YELLOW `#define RGB_YELLOW 0x28`

20.65.2.12 RGB_DARKYELLOW `#define RGB_DARKYELLOW 0x18`

20.65.2.13 RGB_CYAN `#define RGB_CYAN 0x2C`

20.65.2.14 RGB_AQUA `#define RGB_AQUA 0x1C`

20.65.2.15 RGB_PINK `#define RGB_PINK 0x24`

20.65.2.16 RGB_PURPLE `#define RGB_PURPLE 0x14`

20.65.2.17 RGB_BLACK `#define RGB_BLACK 0x0F`

20.65.2.18 RGB_DARKGRAY `#define RGB_DARKGRAY 0x00`

20.65.2.19 RGB_LIGHTGRAY `#define RGB_LIGHTGRAY 0x10`

20.65.2.20 RGB_WHITE `#define RGB_WHITE 0x30`

20.65.2.21 J_UP `#define J_UP 0x10U`

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_t keys; keys = joypad\(\); if (keys & J_B) { ... }
```

See also

[joypad](#)

20.65.2.22 J_DOWN `#define J_DOWN 0x20U`

20.65.2.23 J_LEFT `#define J_LEFT 0x40U`

20.65.2.24 J_RIGHT `#define J_RIGHT 0x80U`

20.65.2.25 J_A `#define J_A 0x01U`

20.65.2.26 J_B `#define J_B 0x02U`

20.65.2.27 J_SELECT `#define J_SELECT 0x04U`

20.65.2.28 J_START `#define J_START 0x08U`

20.65.2.29 M_DRAWING `#define M_DRAWING 0x01U`
Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

20.65.2.30 M_TEXT_OUT `#define M_TEXT_OUT 0x02U`

20.65.2.31 M_TEXT_INOUT `#define M_TEXT_INOUT 0x03U`

20.65.2.32 M_NO_SCROLL `#define M_NO_SCROLL 0x04U`
Set this in addition to the others to disable scrolling
If scrolling is disabled, the cursor returns to (0,0)

See also

[mode\(\)](#)

20.65.2.33 M_NO_INTERP `#define M_NO_INTERP 0x08U`
Set this to disable interpretation

See also

[mode\(\)](#)

20.65.2.34 S_PALETTE `#define S_PALETTE 0x10U`
If this is set, sprite colours come from OBJ1PAL. Else they come from OBJ0PAL

See also

[set_sprite_prop\(\)](#).

20.65.2.35 S_FLIPX `#define S_FLIPX 0x40U`
If set the sprite will be flipped horizontally.

See also

[set_sprite_prop\(\)](#)

20.65.2.36 S_FLIPY `#define S_FLIPY 0x80U`
If set the sprite will be flipped vertically.

See also

[set_sprite_prop\(\)](#)

20.65.2.37 S_PRIORITY `#define S_PRIORITY 0x20U`

If this bit is clear, then the sprite will be displayed on top of the background and window.

See also

[set_sprite_prop\(\)](#)

20.65.2.38 DMG_BLACK `#define DMG_BLACK 0x03`**20.65.2.39 DMG_DARK_GRAY** `#define DMG_DARK_GRAY 0x02`**20.65.2.40 DMG_LITE_GRAY** `#define DMG_LITE_GRAY 0x01`**20.65.2.41 DMG_WHITE** `#define DMG_WHITE 0x00`**20.65.2.42 DMG_PALETTE** `#define DMG_PALETTE(
 C0,
 C1,
 C2,
 C3) ((uint8_t) (((C3) & 0x03) << 6) | (((C2) & 0x03) << 4) | (((C1) & 0x03) <<
2) | ((C0) & 0x03)))`

Macro to create a DMG palette from 4 colors

Parameters

<i>C0</i>	Color for Index 0
<i>C1</i>	Color for Index 1
<i>C2</i>	Color for Index 2
<i>C3</i>	Color for Index 3

The resulting format is four greyscale colors packed into a single unsigned byte.

Example:

```
REG_BGP = DMG_PALETTE(DMG_BLACK, DMG_DARK_GRAY, DMG_LITE_GRAY, DMG_WHITE);
```

See also

[OBP0_REG](#), [OBP1_REG](#), [BGP_REG](#)

[DMG_BLACK](#), [DMG_DARK_GRAY](#), [DMG_LITE_GRAY](#), [DMG_WHITE](#)

20.65.2.43 SCREENWIDTH `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`

Width of the visible screen in pixels.

20.65.2.44 SCREENHEIGHT `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`

Height of the visible screen in pixels.

20.65.2.45 CURRENT_BANK `#define CURRENT_BANK _current_bank`

20.65.2.46 BANK `#define BANK(
VARNAME) 0`

Obtains the **bank number** of VARNAME

Parameters

<i>VARNAME</i>	Name of the variable which has a <code>__bank_</code> <i>VARNAME</i> companion symbol which is adjusted by bankpack
----------------	---

Use this to obtain the bank number from a bank reference created with [BANKREF\(\)](#).

See also

[BANKREF_EXTERN\(\)](#), [BANKREF\(\)](#)

20.65.2.47 BANKREF `#define BANKREF(
 VARNAME)`

Creates a reference for retrieving the bank number of a variable or function

Parameters

<i>VARNAME</i>	Variable name to use, which may be an existing identifier
----------------	---

See also

[BANK\(\)](#) for obtaining the bank number of the included data.

More than one [BANKREF\(\)](#) may be created per file, but each call should always use a unique *VARNAME*. Use [BANKREF_EXTERN\(\)](#) within another source file to make the variable and it's data accesible there.

20.65.2.48 BANKREF_EXTERN `#define BANKREF_EXTERN(
 VARNAME) extern const void __bank_ ## VARNAME;`

Creates extern references for accessing a [BANKREF\(\)](#) generated variable.

Parameters

<i>VARNAME</i>	Name of the variable used with BANKREF()
----------------	--

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

20.65.2.49 SWITCH_ROM_DUMMY `#define SWITCH_ROM_DUMMY(
 b)`

Dummy macro for no-bank-switching WIP prototype

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

20.65.2.50 SWITCH_ROM `#define SWITCH_ROM SWITCH_ROM_DUMMY`

Makes default mapper switch the active ROM bank

Parameters

<i>b</i>	ROM bank to switch to (max 255)
----------	---------------------------------

See also

SWITCH_ROM_UNROM

20.65.2.51 DISPLAY_ON `#define DISPLAY_ON display_on();`

Turns the display back on.

See also

[display_off](#), [DISPLAY_OFF](#)

20.65.2.52 DISPLAY_OFF `#define DISPLAY_OFF display_off();`

Turns the display off immediately.

See also

[display_off](#), [DISPLAY_ON](#)

20.65.2.53 HIDE_LEFT_COLUMN `#define HIDE_LEFT_COLUMN shadow_PPUMASK &= ~(PPUMASK_SHOW_BG_LC | PPUMASK_SHOW_SPR_LC); \`

Blanks leftmost column, so it is not garbaged when you use horizontal scroll

See also

[SHOW_LEFT_COLUMN](#)

20.65.2.54 SHOW_LEFT_COLUMN `#define SHOW_LEFT_COLUMN shadow_PPUMASK |= (PPUMASK_SHOW_BG_LC | PPUMASK_SHOW_SPR_LC);`

Shows leftmost column

See also

[HIDE_LEFT_COLUMN](#)

20.65.2.55 SHOW_BKG `#define SHOW_BKG shadow_PPUMASK |= PPUMASK_SHOW_BG;`

Turns on the background layer. Sets bit 0 of the LCDC register to 1.

20.65.2.56 HIDE_BKG `#define HIDE_BKG shadow_PPUMASK &= ~PPUMASK_SHOW_BG;`

Turns off the background layer. Sets bit 0 of the LCDC register to 0.

20.65.2.57 SHOW_SPRITES `#define SHOW_SPRITES shadow_PPUMASK |= PPUMASK_SHOW_SPR;`

Turns on the sprites layer. Sets bit 1 of the LCDC register to 1.

20.65.2.58 HIDE_SPRITES `#define HIDE_SPRITES shadow_PPUMASK &= ~PPUMASK_SHOW_SPR;`

Turns off the sprites layer. Clears bit 1 of the LCDC register to 0.

20.65.2.59 SPRITES_8x16 `#define SPRITES_8x16 shadow_PPUCTRL |= PPUCTRL_SPR_8X16;`
Sets sprite size to 8x16 pixels, two tiles one above the other. Sets bit 2 of the LCDC register to 1.

20.65.2.60 SPRITES_8x8 `#define SPRITES_8x8 shadow_PPUCTRL &= ~PPUCTRL_SPR_8X16;`
Sets sprite size to 8x8 pixels, one tile. Clears bit 2 of the LCDC register to 0.

20.65.2.61 COMPAT_PALETTE `#define COMPAT_PALETTE(
 C0,
 C1,
 C2,
 C3) ((uint8_t)((C3) << 6) | ((C2) << 4) | ((C1) << 2) | (C0))`

20.65.2.62 set_bkg_2bpp_data `#define set_bkg_2bpp_data set_bkg_data`

20.65.2.63 set_tile_map `#define set_tile_map set_bkg_tiles`

20.65.2.64 set_tile_submap `#define set_tile_submap set_bkg_submap`

20.65.2.65 set_tile_xy `#define set_tile_xy set_bkg_tile_xy`

20.65.2.66 set_sprite_2bpp_data `#define set_sprite_2bpp_data set_sprite_data`

20.65.2.67 DISABLE_OAM_DMA `#define DISABLE_OAM_DMA _shadow_OAM_base = 0`

20.65.2.68 DISABLE_VBL_TRANSFER `#define DISABLE_VBL_TRANSFER DISABLE_OAM_DMA`
Disable OAM DMA copy each VBlank

20.65.2.69 ENABLE_OAM_DMA `#define ENABLE_OAM_DMA _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM
>> 8)`

20.65.2.70 ENABLE_VBL_TRANSFER `#define ENABLE_VBL_TRANSFER ENABLE_OAM_DMA`
Enable OAM DMA copy each VBlank and set it to transfer default shadow_OAM array

20.65.2.71 MAX_HARDWARE_SPRITES `#define MAX_HARDWARE_SPRITES 64`
Amount of hardware sprites in OAM

20.65.2.72 fill_rect `#define fill_rect fill_bkg_rect`

20.65.3 Typedef Documentation

20.65.3.1 palette_color_t `typedef uint8_t palette_color_t`

20.65.3.2 OAM_item_t `typedef struct OAM_item_t OAM_item_t`
Sprite Attributes structure

Parameters

<i>x</i>	X Coordinate of the sprite on screen
<i>y</i>	Y Coordinate of the sprite on screen - 1
<i>tile</i>	Sprite tile number (see set_sprite_tile)
<i>prop</i>	OAM Property Flags (see set_sprite_prop)

20.65.4 Function Documentation

20.65.4.1 set_bkg_palette() `void set_bkg_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

20.65.4.2 set_sprite_palette() `void set_sprite_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

20.65.4.3 set_bkg_palette_entry() `void set_bkg_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `palette_color_t rgb_data)`

20.65.4.4 set_sprite_palette_entry() `void set_sprite_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `palette_color_t rgb_data)`

20.65.4.5 mode() `void mode (`
 `uint8_t m)`

Set the current screen mode - one of M_* modes

Normally used by internal functions only.

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.65.4.6 get_mode() `uint8_t get_mode ()`

Returns the current mode

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.65.4.7 delay() `void delay (`
 `uint16_t d)`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

20.65.4.8 joypad() `uint8_t joypad ()`

Reads and returns the current state of the joypad. Return value is an OR of J_*

When testing for multiple different buttons, it's best to read the joypad state *once* into a variable and then test using that variable.

See also

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

Reads and returns the current state of the joypad. Follows Nintendo's guidelines for reading the pad. Return value is an OR of J_*

When testing for multiple different buttons, it's best to read the joypad state *once* into a variable and then test using that variable.

See also

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

Reads and returns the current state of the joypad.

20.65.4.9 waitpad() `uint8_t waitpad (`
`uint8_t mask)`

Waits until at least one of the buttons 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](#)

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

Waits until at least one of the buttons given in mask are pressed.

Parameters

<i>mask</i>	Bitmask indicating which buttons to wait for
-------------	--

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

Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

See also

[joypad](#)

[J_START](#), [J_SELECT](#), [J_A](#), [J_B](#), [J_UP](#), [J_DOWN](#), [J_LEFT](#), [J_RIGHT](#)

Waits until at least one of the buttons given in mask are pressed.

20.65.4.10 waitpadup() `void waitpadup ()`

Waits for the directional pad and all buttons to be released.

Waits for the directional pad and all buttons to be released.

Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

20.65.4.11 joypad_init() `uint8_t joypad_init (`
`uint8_t npads,`
`joypads_t * joypads)`

Initializes [joypads_t](#) structure for polling multiple joypads

Parameters

<i>npads</i>	number of joypads requested (1, 2 or 4)
<i>joypads</i>	pointer to joypads_t structure to be initialized

Only required for [joypad_ex](#), not required for calls to regular [joypad\(\)](#)

Returns

number of joypads available

See also

[joypad_ex\(\)](#), [joypads_t](#)

20.65.4.12 joypad_ex() `void joypad_ex (
 joypads_t * joypads)`

Polls all available joypads

See also

[joypad_init\(\)](#), [joypads_t](#)

Polls all available joypads (for the GB and ones connected via SGB)

Parameters

<i>joypads</i>	pointer to joypads_t structure to be filled with joypad statuses, must be previously initialized with joypad_init()
----------------	---

See also

[joypad_init\(\)](#), [joypads_t](#)

Polls all available joypads

Parameters

<i>joypads</i>	pointer to joypads_t structure to be filled with joypad statuses, must be previously initialized with joypad_init()
----------------	---

See also

[joypad_init\(\)](#), [joypads_t](#)

20.65.4.13 enable_interrupts() `void enable_interrupts () [inline]`

Enables unmasked interrupts

Note

Use [CRITICAL](#) {...} instead for creating a block of code which should execute with interrupts temporarily turned off.

See also

[disable_interrupts](#), [set_interrupts](#), [CRITICAL](#)

20.65.4.14 disable_interrupts() `void disable_interrupts () [inline]`

Disables interrupts

Note

Use [CRITICAL](#) {...} instead for creating a block of code which should execute with interrupts temporarily turned off.

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](#), [set_interrupts](#), [CRITICAL](#)

20.65.4.15 wait_vbl_done() `void wait_vbl_done ()`

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

This is often used in main loops to idle the CPU until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return.

HALTs the CPU and waits for the vertical blank interrupt (VBL) to finish.

This is often used in main loops to idle the CPU at low power until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediately.

20.65.4.16 display_off() `void display_off () [inline]`

Turns the display off.

Waits until the VBL interrupt before turning the display off.

See also

[DISPLAY_ON](#)

Turns the display off.

See also

[DISPLAY_ON](#)

20.65.4.17 refresh_OAM() `void refresh_OAM ()`

Copies data from shadow OAM to OAM

20.65.4.18 set_vram_byte() `void set_vram_byte (`

`uint8_t * addr,`

`uint8_t v)`

Set byte in vram at given memory location

Parameters

<i>addr</i>	address to write to
<i>v</i>	value

20.65.4.19 get_bkg_xy_addr() `uint8_t* get_bkg_xy_addr (`

`uint8_t x,`

`uint8_t y)`

Get address of X,Y tile of background map

20.65.4.20 set_2bpp_palette() `void set_2bpp_palette (`

```
uint16_t palette ) [inline]
```

Sets palette for 2bpp color translation for GG/SMS, does nothing on GB

20.65.4.21 set_1bpp_colors_ex() void set_1bpp_colors_ex (

```
uint8_t fgcolor,
```

```
uint8_t bgcolor,
```

```
uint8_t mode )
```

20.65.4.22 set_1bpp_colors() void set_1bpp_colors (

```
uint8_t fgcolor,
```

```
uint8_t bgcolor ) [inline]
```

20.65.4.23 set_bkg_data() void set_bkg_data (

```
uint8_t first_tile,
```

```
uint8_t nb_tiles,
```

```
const uint8_t * data )
```

Sets VRAM Tile Pattern data for the Background

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

See also

[set_tile_data](#)

Sets VRAM Tile Pattern data for the Background / Window

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source tile data

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- VBK_REG = [VBK_BANK_0](#) indicates the first bank
- VBK_REG = [VBK_BANK_1](#) indicates the second

See also

[set_win_data](#), [set_tile_data](#)

20.65.4.24 set_bkg_1bpp_data() void set_bkg_1bpp_data (

```
uint8_t first_tile,
```

```
uint8_t nb_tiles,
```

```
const uint8_t * data )
```

Sets VRAM Tile Pattern data for the Background using 1bpp source data

Similar to [set_bkg_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel.

For a given bit that represent a pixel:

- 0 will be expanded into color 0
- 1 will be expanded into color 1, 2 or 3 depending on color argument

See also

[SHOW_BKG](#), [HIDE_BKG](#), [set_bkg_tiles](#)

Sets VRAM Tile Pattern data for the Background / Window using 1bpp source data

Parameters

<i>first_tile</i>	Index of the first Tile to write
<i>nb_tiles</i>	Number of Tiles to write
<i>data</i>	Pointer to (1bpp) source Tile Pattern data

Similar to [set_bkg_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel. For a given bit that represent a pixel:

- 0 will be expanded into the Background color
- 1 will be expanded into the Foreground color

See [set_1bpp_colors](#) for details about setting the Foreground and Background colors.

See also

[SHOW_BKG](#), [HIDE_BKG](#), [set_bkg_tiles](#)
[set_win_1bpp_data](#), [set_sprite_1bpp_data](#)

20.65.4.25 set_bkg_tiles() `void set_bkg_tiles (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * tiles)`

Sets a rectangular region of Background Tile Map.

Entries are copied from map at **tiles** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

Use [set_bkg_submap\(\)](#) instead when:

- Source map is wider than 32 tiles.
- Writing a width that does not match the source map width **and** more than one row high at a time.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See also

[SHOW_BKG](#)
[set_bkg_data](#), [set_bkg_submap](#), [set_win_tiles](#), [set_tiles](#)

Sets a rectangular region of Background Tile Map.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data

Entries are copied from map at **tiles** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

Use [set_bkg_submap\(\)](#) instead when:

- Source map is wider than 32 tiles.
- Writing a width that does not match the source map width **and** more than one row high at a time.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

Note: Patterns 128-255 overlap with patterns 128-255 of the sprite Tile Pattern table.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- [VBK_REG](#) = [VBK_TILES](#) Tile Numbers are written
- [VBK_REG](#) = [VBK_ATTRIBUTES](#) Tile Attributes are written

GBC Tile 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: Flipped Vertically
- Bit 5 - Horizontal flip. Dictates which way up the tile is drawn horizontally.
0: Normal
1: Flipped Horizontally
- 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.

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_submap](#), [set_win_tiles](#), [set_tiles](#)

20.65.4.26 set_bkg_attributes() `void set_bkg_attributes (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * attributes)`

20.65.4.27 set_bkg_based_tiles() `void set_bkg_based_tiles (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * tiles,`
`uint8_t base_tile) [inline]`

Sets a rectangular region of Background Tile Map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>tiles</i>	Pointer to source tile map data
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_bkg_tiles\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_bkg_tiles](#) for more details

20.65.4.28 set_bkg_submap() `void set_bkg_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w) [inline]`

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 32 tiles.

@ param x X Start position in Background Map tile coordinates. Range 0 - 31 @ param y Y Start position in Background Map tile coordinates. Range 0 - 31 @ param w Width of area to set in tiles. Range 1 - 255 @ param h Height of area to set in tiles. Range 1 - 255 @ param map Pointer to source tile map data @ param map_w Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

Use this instead of [set_bkg_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See [set_bkg_tiles](#) for setting CGB attribute maps with **VBK_REG**.

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_tiles](#), [set_win_submap](#), [set_tiles](#)

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 32 tiles.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: $x \& 0x1F$ and $y \& 0x1F$). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: $(map_ptr + x + (y * map_width))$.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: $map_ptr + 1 + (2 * map_width)$

Use this instead of [set_bkg_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See [set_bkg_tiles](#) for setting CGB attribute maps with **VBK_REG**.

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_tiles](#), [set_win_submap](#), [set_tiles](#)

```
20.65.4.29 set_bkg_based_submap() void set_bkg_based_submap (
    uint8_t x,
    uint8_t y,
    uint8_t w,
    uint8_t h,
    const uint8_t * map,
    uint8_t map_w,
    uint8_t base_tile ) [inline]
```

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. The offset value in **base_tile** is added to the tile ID for each map entry.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map_w</i>	Width of source tile map in tiles. Range 1 - 255
<i>base_tile</i>	Offset each tile ID entry of the source map by this value. Range 1 - 255

This is identical to [set_bkg_submap\(\)](#) except that it adds the **base_tile** parameter for when a tile map's tiles don't

start at index zero. (For example, the tiles used by the map range from 100 -> 120 in VRAM instead of 0 -> 20).

See also

[set_bkg_submap](#) for more details

20.65.4.30 get_bkg_tiles() `void get_bkg_tiles (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`uint8_t * tiles)`

Copies a rectangular region of Background Tile Map entries into a buffer.

Entries are copied into **tiles** from the Background Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x** x **y** bytes in size.

See also

[get_bkg_tile_xy](#), [get_tiles](#)

Copies a rectangular region of Background Tile Map entries into a buffer.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Background Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x** x **y** bytes in size.

See also

[get_win_tiles](#), [get_bkg_tile_xy](#), [get_tiles](#), [get_vram_byte](#)

20.65.4.31 set_bkg_tile_xy() `uint8_t* set_bkg_tile_xy (`
`uint8_t x,`
`uint8_t y,`
`uint8_t t)`

Set single tile **t** on background layer at **x**,**y**

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

20.65.4.32 [get_bkg_tile_xy\(\)](#) `uint8_t get_bkg_tile_xy (`
 `uint8_t x,`
 `uint8_t y)`

Get single tile t on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate

Returns

returns tile index

20.65.4.33 [move_bkg\(\)](#) `void move_bkg (`
 `uint8_t x,`
 `uint8_t y) [inline]`

Moves the Background Layer to the position specified in **x** and **y** in pixels.

Parameters

<i>x</i>	X axis screen coordinate for Left edge of the Background
<i>y</i>	Y axis screen coordinate for Top edge of the Background

0,0 is the top left corner of the GB screen. The Background Layer wraps around the screen, so when part of it goes off the screen it appears on the opposite side (factoring in the larger size of the Background Layer versus the screen size).

The background layer is always under the Window Layer.

See also

[SHOW_BKG](#), [HIDE_BKG](#)

20.65.4.34 [scroll_bkg\(\)](#) `void scroll_bkg (`
 `int8_t x,`
 `int8_t y) [inline]`

Moves the Background relative to it's current position.

Parameters

<i>x</i>	Number of pixels to move the Background on the X axis Range: -128 - 127
<i>y</i>	Number of pixels to move the Background on the Y axis Range: -128 - 127

See also

[move_bkg](#)

20.65.4.35 set_sprite_data() `void set_sprite_data (`
 `uint8_t first_tile,`
 `uint8_t nb_tiles,`
 `const uint8_t * data)`

Sets VRAM Tile Pattern data for Sprites

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- VBK_REG=0 indicates the first bank
- VBK_REG=1 indicates the second

Sets VRAM Tile Pattern data for Sprites

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data

Writes **nb_tiles** tiles to VRAM starting at **first_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- VBK_REG = [VBK_BANK_0](#) indicates the first bank
- VBK_REG = [VBK_BANK_1](#) indicates the second

20.65.4.36 set_sprite_1bpp_data() `void set_sprite_1bpp_data (`
 `uint8_t first_tile,`
 `uint8_t nb_tiles,`
 `const uint8_t * data)`

Sets VRAM Tile Pattern data for Sprites using 1bpp source data

Similar to [set_sprite_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel.

For a given bit that represent a pixel:

- 0 will be expanded into color 0
- 1 will be expanded into color 3

See also

[SHOW_SPRITES](#), [HIDE_SPRITES](#), [set_sprite_tile](#)

Sets VRAM Tile Pattern data for Sprites using 1bpp source data

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (1bpp) source Tile Pattern data

Similar to [set_sprite_data](#), except source data is 1 bit-per-pixel which gets expanded into 2 bits-per-pixel. For a given bit that represent a pixel:

- 0 will be expanded into the Background color
- 1 will be expanded into the Foreground color

See [set_1bpp_colors](#) for details about setting the Foreground and Background colors.

See also

[SHOW_SPRITES](#), [HIDE_SPRITES](#), [set_sprite_tile](#)
[set_bkg_1bpp_data](#), [set_win_1bpp_data](#)

20.65.4.37 SET_SHADOW_OAM_ADDRESS() `void SET_SHADOW_OAM_ADDRESS (void * address) [inline]`

Enable OAM DMA copy each VBlank and set it to transfer any 256-byte aligned array

20.65.4.38 set_sprite_tile() `void set_sprite_tile (uint8_t nb, uint8_t tile) [inline]`

Sets sprite number **nb** in the OAM to display tile number **tile**.

Parameters

<i>nb</i>	Sprite number, range 0 - 63
<i>tile</i>	Selects a tile (0 - 255) from PPU memory at 0000h - 0FFFh / 1000h - 1FFFh

In 8x16 mode:

- The sprite will also display the next tile (**tile** + 1) directly below (y + 8) the first tile.
- The lower bit of the tile number is ignored: the upper 8x8 tile is (**tile** & 0xFE), and the lower 8x8 tile is (**tile** | 0x01).
- See: [SPRITES_8x16](#)

20.65.4.39 get_sprite_tile() `uint8_t get_sprite_tile (uint8_t nb) [inline]`

Returns the tile number of sprite number **nb** in the OAM.

Parameters

<i>nb</i>	Sprite number, range 0 - 63
-----------	-----------------------------

See also

[set_sprite_tile](#) for more details

20.65.4.40 set_sprite_prop() `void set_sprite_prop (uint8_t nb, uint8_t prop) [inline]`

Sets the OAM Property Flags of sprite number **nb** to those defined in **prop**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>prop</i>	Property setting (see bitfield description)

The bits in **prop** represent:

- Bit 7 - Priority flag. When this is set the sprites appear behind the background and window layer.
0: infront
1: behind
- Bit 6 - Vertical flip. Dictates which way up the sprite is drawn vertically.
0: normal
1:upside down
- Bit 5 - Horizontal flip. Dictates which way up the sprite is drawn horizontally.
0: normal
1:back to front
- Bit 4 - DMG/Non-CGB Mode 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.

20.65.4.41 get_sprite_prop() `uint8_t get_sprite_prop (`
`uint8_t nb) [inline]`

Returns the OAM Property Flags of sprite number **nb**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[set_sprite_prop](#) for property bitfield settings

20.65.4.42 move_sprite() `void move_sprite (`
`uint8_t nb,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves sprite number **nb** to the **x, y** position on the screen.

Parameters

<i>nb</i>	Sprite number, range 0 - 63
<i>x</i>	X Position. Specifies the sprites horizontal position on the screen (minus 8).
<i>y</i>	Y Position. Specifies the sprites vertical position on the screen (minus 16). An offscreen value ($Y \geq 240$) hides the sprite.

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

20.65.4.43 scroll_sprite() `void scroll_sprite (`
 `uint8_t nb,`
 `int8_t x,`
 `int8_t y) [inline]`

Moves sprite number **nb** relative to its current position.

Parameters

<i>nb</i>	Sprite number, range 0 - 63
<i>x</i>	Number of pixels to move the sprite on the X axis Range: -128 - 127
<i>y</i>	Number of pixels to move the sprite on the Y axis Range: -128 - 127

See also

[move_sprite](#) for more details about the X and Y position

20.65.4.44 hide_sprite() `void hide_sprite (`
 `uint8_t nb) [inline]`

Hides sprite number **nb** by moving it to Y position 240.

Parameters

<i>nb</i>	Sprite number, range 0 - 63
-----------	-----------------------------

20.65.4.45 set_data() `void set_data (`
 `uint8_t * vram_addr,`
 `const uint8_t * data,`
 `uint16_t len)`

Copies arbitrary data to an address in VRAM without taking into account the state of LCDC bits 3 or 4.

Copies **len** bytes from a buffer at **data** to VRAM starting at **vram_addr**.

See also

[set_bkg_data](#), [set_win_data](#), [set_bkg_tiles](#), [set_win_tiles](#), [set_tile_data](#), [set_tiles](#)

Copies arbitrary data to an address in VRAM without taking into account the state of LCDC bits 3 or 4.

Parameters

<i>vram_addr</i>	Pointer to destination VRAM Address
<i>data</i>	Pointer to source buffer
<i>len</i>	Number of bytes to copy

Copies **len** bytes from a buffer at **data** to VRAM starting at **vram_addr**.

GBC only: [VBK_REG](#) determines which bank of Background tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank
- `VBK_REG = VBK_BANK_1` indicates the second

See also

[set_bkg_data](#), [set_win_data](#), [set_bkg_tiles](#), [set_win_tiles](#), [set_tile_data](#), [set_tiles](#)

20.65.4.46 set_tiles() `void set_tiles (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `uint8_t * vram_addr,`
 `const uint8_t * tiles)`

Sets a rectangular region of Tile Map entries at a given VRAM Address.

Parameters

<i>x</i>	X Start position in Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 1 - 32
<i>h</i>	Height of area to set in tiles. Range 1 - 32
<i>vram_addr</i>	Pointer to destination VRAM Address
<i>tiles</i>	Pointer to source Tile Map data

Entries are copied from **tiles** to Tile Map at address *vram_addr* starting at **x**, **y** writing across for **w** tiles and down for **h** tiles.

One byte per source tile map entry.

There are two 32x30 Tile Maps in VRAM at addresses 2000h-23FFh and 2400h-27FFh.

See also

[set_bkg_tiles](#)

20.65.4.47 set_tile_data() `void set_tile_data (`
 `uint8_t first_tile,`
 `uint8_t nb_tiles,`
 `const uint8_t * data,`
 `uint8_t base)`

Sets VRAM Tile Pattern data starting from given base address without taking into account the state of PPUMASK.

See also

[set_bkg_data](#), [set_data](#)

Sets VRAM Tile Pattern data starting from given base address without taking into account the state of LCDC bit 4.

Parameters

<i>first_tile</i>	Index of the first tile to write
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data.
<i>base</i>	MSB of the destination address in VRAM (usually 0x80 or 0x90 which gives 0x8000 or 0x9000)

See also

[set_bkg_data](#), [set_win_data](#), [set_data](#)

20.65.4.48 set_native_tile_data() void set_native_tile_data (
 uint16_t first_tile,
 uint8_t nb_tiles,
 const uint8_t * data) [inline]

Sets VRAM Tile Pattern data in the native format

Parameters

<i>first_tile</i>	Index of the first tile to write (0 - 511)
<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to source Tile Pattern data.

When *first_tile* is larger than 256 on the GB/AP, it will write to sprite data instead of background data. The bit depth of the source Tile Pattern data depends on which console is being used:

- NES: loads 2bpp tiles data

20.65.4.49 init_bkg() void init_bkg (
 uint8_t c)

Initializes the entire Background Tile Map with Tile Number *c*

Parameters

<i>c</i>	Tile number to fill with
----------	--------------------------

Note: This function avoids writes during modes 2 & 3

20.65.4.50 vmemset() void vmemset (
 void * s,
 uint8_t c,
 size_t n)

Fills the VRAM memory region *s* of size *n* with Tile Number *c*

Parameters

<i>s</i>	Start address in VRAM
<i>c</i>	Tile number to fill with
<i>n</i>	Size of memory region (in bytes) to fill

Note: This function avoids writes during modes 2 & 3

20.65.4.51 fill_bkg_rect() void fill_bkg_rect (
 uint8_t x,
 uint8_t y,
 uint8_t w,
 uint8_t h,
 uint8_t tile)

Fills a rectangular region of Tile Map entries for the Background layer with tile.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 0 - 31
<i>h</i>	Height of area to set in tiles. Range 0 - 31

Parameters

<i>tile</i>	Fill value
-------------	------------

20.65.5 Variable Documentation

20.65.5.1 **sys_time** `volatile uint16_t sys_time`

Global Time Counter in VBL periods (60Hz)

Increments once per Frame

Will wrap around every ~18 minutes (unsigned 16 bits = $65535 / 60 / 60 = 18.2$)

20.65.5.2 **_current_bank** `volatile uint8_t _current_bank`

Tracks current active ROM bank

The active bank number is not tracked by `_current_bank` when `SWITCH_ROM_MBC5_8M` is used.

This variable is updated automatically when you call `SWITCH_ROM_MBC1` or `SWITCH_ROM_MBC5`, `SWITCH_ROM()`, or call a BANKED function.

See also

`SWITCH_ROM_MBC1()`, `SWITCH_ROM_MBC5()`, `SWITCH_ROM()`

20.65.5.3 **_current_1bpp_colors** `uint16_t _current_1bpp_colors`

20.65.5.4 **_map_tile_offset** `uint8_t _map_tile_offset`

20.65.5.5 **_submap_tile_offset** `uint8_t _submap_tile_offset`

20.65.5.6 **shadow_OAM** `volatile struct OAM_item_t shadow_OAM[]`

Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

20.65.5.7 **_shadow_OAM_base** `uint8_t _shadow_OAM_base`

MSB of shadow_OAM address is used by OAM DMA copying routine

20.66 gbdk-lib/include/nes/rgb_to_nes_macro.h File Reference

Macros

- `#define RGB_TO_NES(c)`

20.66.1 Macro Definition Documentation

20.66.1.1 **RGB_TO_NES** `#define RGB_TO_NES(c)`

20.67 gbdk-lib/include/rand.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- `#define RAND_MAX 255`
- `#define RANDW_MAX 65535`

Functions

- `void initrand (uint16_t seed) OLDCALL`
- `uint8_t rand () OLDCALL`
- `uint16_t randw () OLDCALL`
- `void initarand (uint16_t seed) OLDCALL`
- `uint8_t arand () OLDCALL`

Variables

- `uint16_t __rand_seed`

20.67.1 Detailed Description

Random generator using the linear congruential method

Author

Luc Van den Borre

20.67.2 Macro Definition Documentation

20.67.2.1 RAND_MAX `#define RAND_MAX 255`

20.67.2.2 RANDW_MAX `#define RANDW_MAX 65535`

20.67.3 Function Documentation

20.67.3.1 initrand() `void initrand (uint16_t seed)`

Initialise the pseudo-random number generator.

Parameters

<code>seed</code>	The value for initializing the random number generator.
-------------------	---

The seed should be different each time, otherwise the same pseudo-random sequence will be generated.

The DIV Register ([DIV_REG](#)) is sometimes used as a seed, particularly if read at some variable point in time (such as when the player presses a button).

Only needs to be called once to initialize, but may be called again to re-initialize with the same or a different seed.

See also

[rand\(\)](#), [randw\(\)](#)

20.67.3.2 rand() `uint8_t rand ()`

Returns a random byte (8 bit) value.

[initrand\(\)](#) should be used to initialize the random number generator before using [rand\(\)](#)

20.67.3.3 randw() `uint16_t randw ()`

Returns a random word (16 bit) value.

`initrand()` should be used to initialize the random number generator before using `rand()`

20.67.3.4 initrand() `void initrand (
uint16_t seed)`

Random generator using the linear lagged additive method

Parameters

<code>seed</code>	The value for initializing the random number generator.
-------------------	---

Note: `initrand()` calls `initrand()` with the same seed value, and uses `rand()` to initialize the random generator.

See also

`initrand()` for suggestions about seed values, `arand()`

20.67.3.5 arand() `uint8_t arand ()`

Returns a random number generated with the linear lagged additive method.

`initrand()` should be used to initialize the random number generator before using `arand()`

20.67.4 Variable Documentation**20.67.4.1 __rand_seed** `uint16_t __rand_seed`

The random number seed is stored in `__rand_seed` and can be saved and restored if needed.

```
// Save
some_uint16 = __rand_seed;
...
// Restore
__rand_seed = some_uint16;
```

20.68 gbdk-lib/include/setjmp.h File Reference**Macros**

- `#define SP_SIZE 1`
- `#define BP_SIZE 0`
- `#define SPX_SIZE 0`
- `#define BPX_SIZE SPX_SIZE`
- `#define RET_SIZE 2`
- `#define setjmp(jump_buf) __setjmp(jump_buf)`

Typedefs

- `typedef unsigned char jmp_buf[RET_SIZE+SP_SIZE+BP_SIZE+SPX_SIZE+BPX_SIZE]`

Functions

- `int __setjmp (jmp_buf) OLDCALL`
- `_Noreturn void longjmp (jmp_buf, int) OLDCALL`

20.68.1 Macro Definition Documentation

20.68.1.1 SP_SIZE `#define SP_SIZE 1`

20.68.1.2 BP_SIZE `#define BP_SIZE 0`

20.68.1.3 SPX_SIZE `#define SPX_SIZE 0`

20.68.1.4 BPX_SIZE `#define BPX_SIZE SPX_SIZE`

20.68.1.5 RET_SIZE `#define RET_SIZE 2`

20.68.1.6 setjmp `#define setjmp(
 jump_buf) __setjmp(jump_buf)`

20.68.2 Typedef Documentation

20.68.2.1 jmp_buf `typedef unsigned char jmp_buf[RET_SIZE+SP_SIZE+BP_SIZE+SPX_SIZE+BPX_SIZE]`

20.68.3 Function Documentation

20.68.3.1 __setjmp() `int __setjmp (
 jmp_buf)`

20.68.3.2 longjmp() `_Noreturn void longjmp (
 jmp_buf ,
 int)`

20.69 gbdk-lib/include/sms/sms.h File Reference

```
#include <types.h>
#include <stdint.h>
#include <gbdk/version.h>
#include <sms/hardware.h>
```

Data Structures

- struct [joypads_t](#)

Macros

- #define [SEGA](#)
- #define [VBK_REG_VDP_ATTR_SHIFT](#)
- #define [J_UP](#) 0b00000001
- #define [J_DOWN](#) 0b00000010
- #define [J_LEFT](#) 0b00000100
- #define [J_RIGHT](#) 0b00001000

```

• #define J_A 0b00010000
• #define J_B 0b00100000
• #define M_TEXT_OUT 0x02U
• #define M_TEXT_INOUT 0x03U
• #define M_NO_SCROLL 0x04U
• #define M_NO_INTERP 0x08U
• #define S_FLIPX 0x02U
• #define S_FLIPY 0x04U
• #define S_PALETTE 0x08U
• #define S_PRIORITY 0x10U
• #define __WRITE_VDP_REG(REG, v) shadow_##REG=(v);__critical{VDP_CMD=(shadow_##REG),VDP←
  _CMD=REG;}
• #define __READ_VDP_REG(REG) shadow_##REG
• #define EMPTY_IFLAG 0x00U
• #define VBL_IFLAG 0x01U
• #define LCD_IFLAG 0x02U
• #define TIM_IFLAG 0x04U
• #define SIO_IFLAG 0x08U
• #define JOY_IFLAG 0x10U
• #define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH
• #define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT
• #define MINWNDPOSX 0x00U
• #define MINWNDPOSY 0x00U
• #define MAXWNDPOSX 0x00U
• #define MAXWNDPOSY 0x00U
• #define DISPLAY_ON __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) |= R1_DISP_ON)
• #define DISPLAY_OFF display_off();
• #define HIDE_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) |= R0_LCB)
• #define SHOW_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) &=
  (~R0_LCB))
• #define SHOW_BKG
• #define HIDE_BKG
• #define SHOW_WIN
• #define HIDE_WIN
• #define SHOW_SPRITES
• #define HIDE_SPRITES
• #define SPRITES_8x16 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) |= R1_SPR_8X16)
• #define SPRITES_8x8 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) &= (~R1_SPR_8X16))
• #define DEVICE_SUPPORTS_COLOR (TRUE)
• #define _current_bank MAP_FRAME1
• #define CURRENT_BANK MAP_FRAME1
• #define BANK(VARNAME) ( (uint8_t) & __bank_ ## VARNAME )
• #define BANKREF(VARNAME)
• #define BANKREF_EXTERN(VARNAME) extern const void __bank_ ## VARNAME;
• #define SWITCH_ROM(b) MAP_FRAME1=(b)
• #define SWITCH_ROM1 SWITCH_ROM
• #define SWITCH_ROM2(b) MAP_FRAME2=(b)
• #define SWITCH_RAM(b) RAM_CONTROL=((b)&1)?RAM_CONTROL|RAMCTL_BANK:RAM_CONTR←
  OL&(~RAMCTL_BANK)
• #define ENABLE_RAM RAM_CONTROL|=RAMCTL_RAM
• #define DISABLE_RAM RAM_CONTROL&= (~RAMCTL_RAM)
• #define set_bkg_palette_entry set_palette_entry
• #define set_sprite_palette_entry(palette, entry, rgb_data) set_palette_entry(1,entry,rgb_data)
• #define set_bkg_palette set_palette
• #define set_sprite_palette(first_palette, nb_palettes, rgb_data) set_palette(1,1,rgb_data)

```

- `#define COMPAT_PALETTE(C0, C1, C2, C3) (((uint16_t)(C3) << 12) | ((uint16_t)(C2) << 8) | ((uint16_t)(C1) << 4) | (uint16_t)(C0))`
- `#define set_bkg_tiles set_tile_map_compat`
- `#define set_win_tiles set_tile_map_compat`
- `#define fill_bkg_rect fill_rect_compat`
- `#define fill_win_rect fill_rect_compat`
- `#define DISABLE_VBL_TRANSFER _shadow_OAM_base = 0`
- `#define ENABLE_VBL_TRANSFER _shadow_OAM_base = (uint8_t)((uint16_t)&shadow_OAM >> 8)`
- `#define MAX_HARDWARE_SPRITES 64`
- `#define set_bkg_tile_xy set_tile_xy`
- `#define set_win_tile_xy set_tile_xy`
- `#define get_win_xy_addr get_bkg_xy_addr`

Typedefs

- `typedef void(* int_handler) (void) NONBANKED`

Functions

- `void WRITE_VDP_CMD (uint16_t cmd) Z88DK_FASTCALL PRESERVES_REGS(b)`
- `void WRITE_VDP_DATA (uint16_t data) Z88DK_FASTCALL PRESERVES_REGS(b)`
- `void mode (uint8_t m) OLDCALL`
- `uint8_t get_mode () OLDCALL`
- `void set_interrupts (uint8_t flags) Z88DK_FASTCALL`
- `void remove_VBL (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(iyh)`
- `void remove_LCD (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(b)`
- `void remove_TIM (int_handler h) Z88DK_FASTCALL`
- `void remove_SIO (int_handler h) Z88DK_FASTCALL`
- `void remove_JOY (int_handler h) Z88DK_FASTCALL`
- `void add_VBL (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(d)`
- `void add_LCD (int_handler h) Z88DK_FASTCALL PRESERVES_REGS(b)`
- `void add_TIM (int_handler h) Z88DK_FASTCALL`
- `void add_SIO (int_handler h) Z88DK_FASTCALL`
- `void add_JOY (int_handler h) Z88DK_FASTCALL`
- `uint8_t cancel_pending_interrupts ()`
- `void move_bkg (uint8_t x, uint8_t y)`
- `void scroll_bkg (int8_t x, int8_t y)`
- `void wait_vbl_done () PRESERVES_REGS(b)`
- `void display_off ()`
- `void refresh_OAM ()`
- `void delay (uint16_t d) Z88DK_FASTCALL`
- `uint8_t joypad () OLDCALL PRESERVES_REGS(b)`
- `uint8_t waitpad (uint8_t mask) Z88DK_FASTCALL PRESERVES_REGS(b)`
- `void waitpadup () PRESERVES_REGS(b)`
- `uint8_t joypad_init (uint8_t npads, joypads_t *joypads) Z88DK_CALLEE`
- `void joypad_ex (joypads_t *joypads) Z88DK_FASTCALL PRESERVES_REGS(iyh)`
- `void set_default_palette ()`
- `void cgb_compatibility ()`
- `void cpu_fast ()`
- `void set_palette_entry (uint8_t palette, uint8_t entry, uint16_t rgb_data) Z88DK_CALLEE PRESERVES_REGS(iyh)`
- `void set_palette (uint8_t first_palette, uint8_t nb_palettes, palette_color_t *rgb_data) Z88DK_CALLEE`
- `void set_native_tile_data (uint16_t start, uint16_t ntiles, const void *src) Z88DK_CALLEE PRESERVES_REGS(iyh)`
- `void set_bkg_4bpp_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_sprite_4bpp_data (uint16_t start, uint16_t ntiles, const void *src)`
- `void set_2bpp_palette (uint16_t palette)`

- void `set_tile_2bpp_data` (uint16_t start, uint16_t ntiles, const void *src, uint16_t palette) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_sprite_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_bkg_2bpp_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_sprite_2bpp_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_1bpp_colors` (uint8_t fgcolor, uint8_t bgcolor)
- void `set_tile_1bpp_data` (uint16_t start, uint16_t ntiles, const void *src, uint16_t colors) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_1bpp_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_sprite_1bpp_data` (uint16_t start, uint16_t ntiles, const void *src)
- void `set_data` (uint16_t dst, const void *src, uint16_t size) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `vmemcpy` (uint16_t dst, const void *src, uint16_t size) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_tile_map` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_tile_map_compat` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_based_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void `set_win_based_tiles` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *tiles, uint8_t base_tile)
- void `set_tile_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t map_w, const uint8_t *map) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_tile_submap_compat` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t map_w, const uint8_t *map) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `set_bkg_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w)
- void `set_win_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w)
- void `set_bkg_based_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void `set_win_based_submap` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint8_t *map, uint8_t map_w, uint8_t base_tile)
- void `fill_rect` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint16_t tile) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `fill_rect_compat` (uint8_t x, uint8_t y, uint8_t w, uint8_t h, const uint16_t tile) Z88DK_CALLEE PRESERVES_REGS(iyh)
- void `SET_SHADOW_OAM_ADDRESS` (void *address)
- void `set_sprite_tile` (uint8_t nb, uint8_t tile)
- uint8_t `get_sprite_tile` (uint8_t nb)
- void `set_sprite_prop` (uint8_t nb, uint8_t prop)
- uint8_t `get_sprite_prop` (uint8_t nb)
- void `move_sprite` (uint8_t nb, uint8_t x, uint8_t y)
- void `scroll_sprite` (uint8_t nb, int8_t x, int8_t y)
- void `hide_sprite` (uint8_t nb)
- void `set_vram_byte` (uint8_t *addr, uint8_t v) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `set_attributed_tile_xy` (uint8_t x, uint8_t y, uint16_t t) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `set_tile_xy` (uint8_t x, uint8_t y, uint8_t t) Z88DK_CALLEE PRESERVES_REGS(iyh)
- uint8_t * `get_bkg_xy_addr` (uint8_t x, uint8_t y) Z88DK_CALLEE PRESERVES_REGS(iyh)

Variables

- void `c`
- void `d`
- void `e`
- void `iyh`
- void `iyi`
- void `h`
- void `l`
- volatile uint16_t `sys_time`

- [uint16_t _current_2bpp_palette](#)
- [uint16_t _current_1bpp_colors](#)
- [uint8_t _map_tile_offset](#)
- [uint8_t _submap_tile_offset](#)
- volatile [uint8_t shadow_OAM \[\]](#)
- volatile [uint8_t _shadow_OAM_base](#)
- volatile [uint8_t _shadow_OAM_OFF](#)

20.69.1 Detailed Description

SMS/GG specific functions.

20.69.2 Macro Definition Documentation

20.69.2.1 SEGA `#define SEGA`

20.69.2.2 VBK_REG `#define VBK_REG VDP_ATTR_SHIFT`

20.69.2.3 J_UP `#define J_UP 0b00000001`

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_t keys; keys = joypad(); if (keys & J_B) { ... }
```

See also

[joypad](#)

20.69.2.4 J_DOWN `#define J_DOWN 0b00000010`

20.69.2.5 J_LEFT `#define J_LEFT 0b00000100`

20.69.2.6 J_RIGHT `#define J_RIGHT 0b00001000`

20.69.2.7 J_A `#define J_A 0b00010000`

20.69.2.8 J_B `#define J_B 0b00100000`

20.69.2.9 M_TEXT_OUT `#define M_TEXT_OUT 0x02U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

20.69.2.10 M_TEXT_INOUT `#define M_TEXT_INOUT 0x03U`

20.69.2.11 M_NO_SCROLL `#define M_NO_SCROLL 0x04U`

Set this in addition to the others to disable scrolling

If scrolling is disabled, the cursor returns to (0,0)

See also

[mode\(\)](#)

20.69.2.12 M_NO_INTERP `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

20.69.2.13 S_FLIPX `#define S_FLIPX 0x02U`

If set the background tile will be flipped horizontally.

20.69.2.14 S_FLIPY `#define S_FLIPY 0x04U`

If set the background tile will be flipped vertically.

20.69.2.15 S_PALETTE `#define S_PALETTE 0x08U`

If set the background tile palette.

20.69.2.16 S_PRIORITY `#define S_PRIORITY 0x10U`

If set the background tile priority.

20.69.2.17 __WRITE_VDP_REG `#define __WRITE_VDP_REG(
 REG,
 v) shadow_##REG=(v);__critical{VDP_CMD=(shadow_##REG),VDP_CMD=REG;}`

20.69.2.18 __READ_VDP_REG `#define __READ_VDP_REG(
 REG) shadow_##REG`

20.69.2.19 EMPTY_IFLAG `#define EMPTY_IFLAG 0x00U`

Disable calling of interrupt service routines

20.69.2.20 VBL_IFLAG `#define VBL_IFLAG 0x01U`

VBlank Interrupt occurs at the start of the vertical blank.

During this period the video ram may be freely accessed.

See also

[set_interrupts\(\)](#),

[add_VBL](#)

20.69.2.21 LCD_IFLAG `#define LCD_IFLAG 0x02U`
LCD Interrupt when triggered by the STAT register.

See also

[set_interrupts\(\)](#),
[add_LCD](#)

20.69.2.22 TIM_IFLAG `#define TIM_IFLAG 0x04U`
Does nothing on SMS/GG

20.69.2.23 SIO_IFLAG `#define SIO_IFLAG 0x08U`
Does nothing on SMS/GG

20.69.2.24 JOY_IFLAG `#define JOY_IFLAG 0x10U`
Does nothing on SMS/GG

20.69.2.25 SCREENWIDTH `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`
Width of the visible screen in pixels.

20.69.2.26 SCREENHEIGHT `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`
Height of the visible screen in pixels.

20.69.2.27 MINWNDPOSX `#define MINWNDPOSX 0x00U`
The Minimum X position of the Window Layer (Left edge of screen)

See also

[move_win\(\)](#)

20.69.2.28 MINWNDPOSY `#define MINWNDPOSY 0x00U`
The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move_win\(\)](#)

20.69.2.29 MAXWNDPOSX `#define MAXWNDPOSX 0x00U`
The Maximum X position of the Window Layer (Right edge of screen)

See also

[move_win\(\)](#)

20.69.2.30 MAXWNDPOSY `#define MAXWNDPOSY 0x00U`
The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move_win\(\)](#)

20.69.2.31 DISPLAY_ON `#define DISPLAY_ON __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) | R1_DISP_ON)`

Turns the display back on.

See also

[display_off](#), [DISPLAY_OFF](#)

20.69.2.32 DISPLAY_OFF `#define DISPLAY_OFF display_off();`

Turns the display off immediately.

See also

[display_off](#), [DISPLAY_ON](#)

20.69.2.33 HIDE_LEFT_COLUMN `#define HIDE_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) | R0_LCB)`

Blanks leftmost column, so it is not garbaged when you use horizontal scroll

See also

[SHOW_LEFT_COLUMN](#)

20.69.2.34 SHOW_LEFT_COLUMN `#define SHOW_LEFT_COLUMN __WRITE_VDP_REG(VDP_R0, __READ_VDP_REG(VDP_R0) &= (~R0_LCB))`

Shows leftmost column

See also

[HIDE_LEFT_COLUMN](#)

20.69.2.35 SHOW_BKG `#define SHOW_BKG`

Turns on the background layer. Not yet implemented

20.69.2.36 HIDE_BKG `#define HIDE_BKG`

Turns off the background layer. Not yet implemented

20.69.2.37 SHOW_WIN `#define SHOW_WIN`

Turns on the window layer. Not yet implemented

20.69.2.38 HIDE_WIN `#define HIDE_WIN`

Turns off the window layer. Not yet implemented

20.69.2.39 SHOW_SPRITES `#define SHOW_SPRITES`

Turns on the sprites layer. Not yet implemented

20.69.2.40 HIDE_SPRITES `#define HIDE_SPRITES`

Turns off the sprites layer. Not yet implemented

20.69.2.41 SPRITES_8x16 `#define SPRITES_8x16 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) | R1_SPR_8X16)`

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

20.69.2.42 SPRITES_8x8 `#define SPRITES_8x8 __WRITE_VDP_REG(VDP_R1, __READ_VDP_REG(VDP_R1) &= (~R1_SPR_8X16))`

Sets sprite size to 8x8 pixels, one tile.

20.69.2.43 DEVICE_SUPPORTS_COLOR `#define DEVICE_SUPPORTS_COLOR (TRUE)`

Macro returns TRUE if device supports color (it always does on SMS/GG)

20.69.2.44 _current_bank `#define _current_bank MAP_FRAME1`

Tracks current active ROM bank in frame 1

20.69.2.45 CURRENT_BANK `#define CURRENT_BANK MAP_FRAME1`

20.69.2.46 BANK `#define BANK(VARNAME) ((uint8_t) & __bank_ ## VARNAME)`

Obtains the **bank number** of VARNAME

Parameters

<i>VARNAME</i>	Name of the variable which has a <code>__bank_</code> <i>VARNAME</i> companion symbol which is adjusted by bankpack
----------------	---

Use this to obtain the bank number from a bank reference created with [BANKREF\(\)](#).

See also

[BANKREF_EXTERN\(\)](#), [BANKREF\(\)](#)

20.69.2.47 BANKREF `#define BANKREF(VARNAME)`

Value:

```
void __func_ ## VARNAME() __banked __naked { \
__asm \
    .local b__func_ ## VARNAME \
    __bank_ ## VARNAME = b__func_ ## VARNAME \
    .globl __bank_ ## VARNAME \
__endasm; \
}
```

Creates a reference for retrieving the bank number of a variable or function

Parameters

<i>VARNAME</i>	Variable name to use, which may be an existing identifier
----------------	---

See also

[BANK\(\)](#) for obtaining the bank number of the included data.

More than one [BANKREF \(\)](#) may be created per file, but each call should always use a unique VARNAME.

Use [BANKREF_EXTERN\(\)](#) within another source file to make the variable and it's data accesible there.

20.69.2.48 BANKREF_EXTERN `#define BANKREF_EXTERN(VARNAME) extern const void __bank_ ## VARNAME;`

Creates extern references for accessing a [BANKREF\(\)](#) generated variable.

Parameters

<i>VARNAME</i>	Name of the variable used with BANKREF()
----------------	--

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

20.69.2.49 SWITCH_ROM `#define SWITCH_ROM(
 b) MAP_FRAME1=(b)`

Makes switch the active ROM bank in frame 1

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

20.69.2.50 SWITCH_ROM1 `#define SWITCH_ROM1 SWITCH_ROM`

20.69.2.51 SWITCH_ROM2 `#define SWITCH_ROM2(
 b) MAP_FRAME2=(b)`

Makes switch the active ROM bank in frame 2

Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

20.69.2.52 SWITCH_RAM `#define SWITCH_RAM(
 b) RAM_CONTROL=((b) & 1) ? RAM_CONTROL | RAMCTL_BANK : RAM_CONTROL & (~RAMCTL_BANK)`

Switches RAM bank

Parameters

<i>b</i>	SRAM bank to switch to
----------	------------------------

20.69.2.53 ENABLE_RAM `#define ENABLE_RAM RAM_CONTROL|=RAMCTL_RAM`
Enables RAM

20.69.2.54 DISABLE_RAM `#define DISABLE_RAM RAM_CONTROL&=(~RAMCTL_RAM)`
Disables RAM

20.69.2.55 set_bkg_palette_entry `#define set_bkg_palette_entry set_palette_entry`

20.69.2.56 set_sprite_palette_entry `#define set_sprite_palette_entry(
 palette,
 entry,
 rgb_data) set_palette_entry(1, entry, rgb_data)`

20.69.2.57 set_bkg_palette #define set_bkg_palette [set_palette](#)

20.69.2.58 set_sprite_palette #define set_sprite_palette(
 first_palette,
 nb_palettes,
 rgb_data) [set_palette](#)(1,1,rgb_data)

20.69.2.59 COMPAT_PALETTE #define COMPAT_PALETTE(
 C0,
 C1,
 C2,
 C3) (([uint16_t](#)) (*C3*) << 12) | (([uint16_t](#)) (*C2*) << 8) | (([uint16_t](#)) (*C1*) << 4) |
 ([uint16_t](#)) (*C0*))

20.69.2.60 set_bkg_tiles #define set_bkg_tiles [set_tile_map_compat](#)

20.69.2.61 set_win_tiles #define set_win_tiles [set_tile_map_compat](#)

20.69.2.62 fill_bkg_rect #define fill_bkg_rect [fill_rect_compat](#)

20.69.2.63 fill_win_rect #define fill_win_rect [fill_rect_compat](#)

20.69.2.64 DISABLE_VBL_TRANSFER #define DISABLE_VBL_TRANSFER [_shadow_OAM_base](#) = 0
 Disable shadow OAM to VRAM copy on each VBlank

20.69.2.65 ENABLE_VBL_TRANSFER #define ENABLE_VBL_TRANSFER [_shadow_OAM_base](#) = ([uint8_t](#)) (([uint16_t](#)) &[shadow_OAM_base](#) >> 8)
 Enable shadow OAM to VRAM copy on each VBlank

20.69.2.66 MAX_HARDWARE_SPRITES #define MAX_HARDWARE_SPRITES 64
 Amount of hardware sprites in OAM

20.69.2.67 set_bkg_tile_xy #define set_bkg_tile_xy [set_tile_xy](#)

20.69.2.68 set_win_tile_xy #define set_win_tile_xy [set_tile_xy](#)

20.69.2.69 get_win_xy_addr #define get_win_xy_addr [get_bkg_xy_addr](#)

20.69.3 Typedef Documentation

20.69.3.1 int_handler typedef void(* int_handler) (void) [NONBANKED](#)
 Interrupt handlers

20.69.4 Function Documentation

20.69.4.1 WRITE_VDP_CMD() `void WRITE_VDP_CMD (`
`uint16_t cmd)`

20.69.4.2 WRITE_VDP_DATA() `void WRITE_VDP_DATA (`
`uint16_t data)`

20.69.4.3 mode() `void mode (`
`uint8_t m)`

Set the current screen mode - one of M_* modes
Normally used by internal functions only.

See also

[M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.69.4.4 get_mode() `uint8_t get_mode ()`
Returns the current mode

See also

[M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

Returns the current mode

See also

[M_DRAWING](#), [M_TEXT_OUT](#), [M_TEXT_INOUT](#), [M_NO_SCROLL](#), [M_NO_INTERP](#)

20.69.4.5 set_interrupts() `void set_interrupts (`
`uint8_t flags)`

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

Parameters

<i>flags</i>	A logical OR of *_IFLAGS
--------------	--------------------------

Note

: This disables and then re-enables interrupts so it must be used outside of a critical section.

See also

[enable_interrupts\(\)](#), [disable_interrupts\(\)](#)
[VBL_IFLAG](#), [LCD_IFLAG](#), [TIM_IFLAG](#), [SIO_IFLAG](#), [JOY_IFLAG](#)

20.69.4.6 remove_VBL() `void remove_VBL (`
`int_handler h)`

Removes the VBL interrupt handler.

See also

[add_VBL\(\)](#)

20.69.4.7 remove_LCD() `void remove_LCD (`
 `int_handler h)`

Removes the LCD interrupt handler.

See also

[add_LCD\(\)](#), [remove_VBL\(\)](#)

20.69.4.8 remove_TIM() `void remove_TIM (`
 `int_handler h)`

20.69.4.9 remove_SIO() `void remove_SIO (`
 `int_handler h)`

20.69.4.10 remove_JOY() `void remove_JOY (`
 `int_handler h)`

20.69.4.11 add_VBL() `void add_VBL (`
 `int_handler h)`

Adds a V-blank interrupt handler.

20.69.4.12 add_LCD() `void add_LCD (`
 `int_handler h)`

Adds a LCD interrupt handler.

20.69.4.13 add_TIM() `void add_TIM (`
 `int_handler h)`

Does nothing on SMS/GG

20.69.4.14 add_SIO() `void add_SIO (`
 `int_handler h)`

Does nothing on SMS/GG

20.69.4.15 add_JOY() `void add_JOY (`
 `int_handler h)`

Does nothing on SMS/GG

20.69.4.16 cancel_pending_interrupts() `uint8_t cancel_pending_interrupts () [inline]`
Cancel pending interrupts

20.69.4.17 move_bkg() `void move_bkg (`
 `uint8_t x,`
 `uint8_t y) [inline]`

20.69.4.18 scroll_bkg() `void scroll_bkg (`
 `int8_t x,`
 `int8_t y) [inline]`

20.69.4.19 wait_vbl_done() `void wait_vbl_done ()`

HALTs the CPU and waits for the vertical blank interrupt (VBL) to finish.

This is often used in main loops to idle the CPU at low power until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediately.

20.69.4.20 display_off() `void display_off () [inline]`

Turns the display off.

See also

[DISPLAY_ON](#)

20.69.4.21 refresh_OAM() `void refresh_OAM ()`

Copies data from shadow OAM to OAM

20.69.4.22 delay() `void delay (`
 `uint16_t d)`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

20.69.4.23 joyypad() `uint8_t joyypad ()`

Reads and returns the current state of the joyypad.

20.69.4.24 waitpad() `uint8_t waitpad (`
 `uint8_t mask)`

Waits until at least one of the buttons given in mask are pressed.

20.69.4.25 waitpadup() `void waitpadup ()`

Waits for the directional pad and all buttons to be released.

Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

20.69.4.26 joyypad_init() `uint8_t joyypad_init (`
 `uint8_t npads,`
 `joypads_t * joypads)`

Initializes [joypads_t](#) structure for polling multiple joypads

Parameters

<i>npads</i>	number of joypads requested (1, 2 or 4)
<i>joypads</i>	pointer to joypads_t structure to be initialized

Only required for [joypad_ex](#), not required for calls to regular [joyypad\(\)](#)

Returns

number of joypads available

See also

[joypad_ex\(\)](#), [joypads_t](#)

20.69.4.27 joypad_ex() `void joypad_ex (`
 `joypads_t * joypads)`

Polls all available joypads

Parameters

<code>joypads</code>	pointer to <code>joypads_t</code> structure to be filled with joypad statuses, must be previously initialized with <code>joypad_init()</code>
----------------------	---

See also

[joypad_init\(\)](#), [joypads_t](#)

20.69.4.28 set_default_palette() `void set_default_palette ()`

20.69.4.29 cgb_compatibility() `void cgb_compatibility ()` `[inline]`

This function is obsolete

20.69.4.30 cpu_fast() `void cpu_fast ()` `[inline]`

Set CPU speed to fast (CGB Double Speed) operation.

On startup the CGB operates in Normal Speed Mode and can be switched into Double speed mode (faster processing but also higher power consumption). See the Pan Docs for more information about which hardware features operate faster and which remain at Normal Speed.

- Interrupts are temporarily disabled and then re-enabled during this call.
- You can check to see if `_cpu == CGB_TYPE` before using this function.

See also

[cpu_slow\(\)](#), [_cpu](#)

20.69.4.31 set_palette_entry() `void set_palette_entry (`
 `uint8_t palette,`
 `uint8_t entry,`
 `uint16_t rgb_data)`

20.69.4.32 set_palette() `void set_palette (`
 `uint8_t first_palette,`
 `uint8_t nb_palettes,`
 `palette_color_t * rgb_data)`

20.69.4.33 set_native_tile_data() `void set_native_tile_data (`
 `uint16_t start,`
 `uint16_t ntiles,`
 `const void * src)`

20.69.4.34 set_bkg_4bpp_data() `void set_bkg_4bpp_data (`
 `uint16_t start,`
 `uint16_t ntiles,`
 `const void * src)` `[inline]`

20.69.4.35 set_sprite_4bpp_data() void set_sprite_4bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.36 set_2bpp_palette() void set_2bpp_palette (
 uint16_t palette) [inline]

20.69.4.37 set_tile_2bpp_data() void set_tile_2bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src,
 uint16_t palette)

20.69.4.38 set_bkg_data() void set_bkg_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.39 set_sprite_data() void set_sprite_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.40 set_bkg_2bpp_data() void set_bkg_2bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.41 set_sprite_2bpp_data() void set_sprite_2bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.42 set_1bpp_colors() void set_1bpp_colors (
 uint8_t fgcolor,
 uint8_t bgcolor) [inline]

20.69.4.43 set_tile_1bpp_data() void set_tile_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src,
 uint16_t colors)

20.69.4.44 set_bkg_1bpp_data() void set_bkg_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.45 set_sprite_1bpp_data() void set_sprite_1bpp_data (
 uint16_t start,
 uint16_t ntiles,
 const void * src) [inline]

20.69.4.46 set_data() void set_data (
 uint16_t dst,
 const void * src,
 uint16_t size)

Copies arbitrary data to an address in VRAM

Parameters

<i>dst</i>	destination VRAM Address
<i>src</i>	Pointer to source buffer
<i>size</i>	Number of bytes to copy

Copies **size** bytes from a buffer at **_src__** to VRAM starting at **dst**.

20.69.4.47 vmemcpy() void vmemcpy (
 uint16_t dst,
 const void * src,
 uint16_t size)

20.69.4.48 set_tile_map() void set_tile_map (
 uint8_t x,
 uint8_t y,
 uint8_t w,
 uint8_t h,
 const uint8_t * tiles)

20.69.4.49 set_tile_map_compat() void set_tile_map_compat (
 uint8_t x,
 uint8_t y,
 uint8_t w,
 uint8_t h,
 const uint8_t * tiles)

20.69.4.50 set_bkg_based_tiles() void set_bkg_based_tiles (
 uint8_t x,
 uint8_t y,
 uint8_t w,
 uint8_t h,
 const uint8_t * tiles,
 uint8_t base_tile) [inline]

20.69.4.51 set_win_based_tiles() void set_win_based_tiles (

```

uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * tiles,
uint8_t base_tile ) [inline]

```

20.69.4.52 set_tile_submap() void set_tile_submap (

```

uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
uint8_t map_w,
const uint8_t * map )

```

20.69.4.53 set_tile_submap_compat() void set_tile_submap_compat (

```

uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
uint8_t map_w,
const uint8_t * map )

```

20.69.4.54 set_bkg_submap() void set_bkg_submap (

```

uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * map,
uint8_t map_w ) [inline]

```

Sets a rectangular area of the Background Tile Map using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 32 tiles.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Background Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: $x \& 0x1F$ and $y \& 0x1F$). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: $(map_ptr + x + (y * map_width))$.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map

(instead of at 1, 2) then modify the pointer address that is passed in: `map_ptr + 1 + (2 * map_width)`
 Use this instead of [set_bkg_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

See [set_bkg_tiles](#) for setting CGB attribute maps with [VBK_REG](#).

See also

[SHOW_BKG](#)

[set_bkg_data](#), [set_bkg_tiles](#), [set_win_submap](#), [set_tiles](#)

20.69.4.55 set_win_submap() `void set_win_submap (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t w,`
 `uint8_t h,`
 `const uint8_t * map,`
 `uint8_t map_w) [inline]`

Sets a rectangular area of the Window Tile Map using a sub-region from a source tile map.

Parameters

<i>x</i>	X Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>y</i>	Y Start position in both the Source Tile Map and hardware Window Map tile coordinates. Range 0 - 255
<i>w</i>	Width of area to set in tiles. Range 1 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map↔ _w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Window Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map_w** as the rowstride for the source tile map.

The **x** and **y** parameters are in Source Tile Map tile coordinates. The location tiles will be written to on the hardware Background Map is derived from those, but only uses the lower 5 bits of each axis, for range of 0-31 (they are bit-masked: `x & 0x1F` and `y & 0x1F`). As a result the two coordinate systems are aligned together.

In order to transfer tile map data in a way where the coordinate systems are not aligned, an offset from the Source Tile Map pointer can be passed in: `(map_ptr + x + (y * map_width))`.

For example, if you want the tile id at 1, 2 from the source map to show up at 0, 0 on the hardware Background Map (instead of at 1, 2) then modify the pointer address that is passed in: `map_ptr + 1 + (2 * map_width)`

Use this instead of [set_win_tiles](#) when the source map is wider than 32 tiles or when writing a width that does not match the source map width.

One byte per source tile map entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

GBC only: [VBK_REG](#) determines whether Tile Numbers or Tile Attributes get set.

- `VBK_REG = VBK_TILES` Tile Numbers are written
- `VBK_REG = VBK_ATTRIBUTES` Tile Attributes are written

See [set_bkg_tiles](#) for details about CGB attribute maps with [VBK_REG](#).

See also

[SHOW_WIN](#), [HIDE_WIN](#), [set_win_tiles](#), [set_bkg_submap](#), [set_bkg_tiles](#), [set_bkg_data](#), [set_tiles](#)

20.69.4.56 set_bkg_based_submap() `void set_bkg_based_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w,`
`uint8_t base_tile) [inline]`

20.69.4.57 set_win_based_submap() `void set_win_based_submap (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint8_t * map,`
`uint8_t map_w,`
`uint8_t base_tile) [inline]`

20.69.4.58 fill_rect() `void fill_rect (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint16_t tile)`

20.69.4.59 fill_rect_compat() `void fill_rect_compat (`
`uint8_t x,`
`uint8_t y,`
`uint8_t w,`
`uint8_t h,`
`const uint16_t tile)`

20.69.4.60 SET_SHADOW_OAM_ADDRESS() `void SET_SHADOW_OAM_ADDRESS (`
`void * address) [inline]`

Sets address of 256-byte aligned array of shadow OAM to be transferred on each VBlank

20.69.4.61 set_sprite_tile() `void set_sprite_tile (`
`uint8_t nb,`
`uint8_t tile) [inline]`

Sets sprite number **nb** in the OAM to display tile number **tile**.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>tile</i>	Selects a tile (0 - 255) from memory at 8000h - 8FFFh In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the OAM Attribute Flag (see set_sprite_prop)

In 8x16 mode:

- The sprite will also display the next tile (**tile** + 1) directly below (y + 8) the first tile.
- The lower bit of the tile number is ignored: the upper 8x8 tile is (**tile** & 0xFE), and the lower 8x8 tile is (**tile** |

0x01).

- See: [SPRITES_8x16](#)

20.69.4.62 get_sprite_tile() `uint8_t get_sprite_tile (`
`uint8_t nb) [inline]`

Returns the tile number of sprite number **nb** in the OAM.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

See also

[set_sprite_tile](#) for more details

20.69.4.63 set_sprite_prop() `void set_sprite_prop (`
`uint8_t nb,`
`uint8_t prop) [inline]`

20.69.4.64 get_sprite_prop() `uint8_t get_sprite_prop (`
`uint8_t nb) [inline]`

20.69.4.65 move_sprite() `void move_sprite (`
`uint8_t nb,`
`uint8_t x,`
`uint8_t y) [inline]`

Moves sprite number **nb** to the **x, y** position on the screen.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	X Position. Specifies the sprites horizontal position on the screen (minus 8). An offscreen value (X=0 or X>=168) hides the sprite, but the sprite still affects the priority ordering - a better way to hide a sprite is to set its Y-coordinate offscreen.
<i>y</i>	Y Position. Specifies the sprites vertical position on the screen (minus 16). An offscreen value (for example, Y=0 or Y>=160) hides the sprite.

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

20.69.4.66 scroll_sprite() `void scroll_sprite (`
`uint8_t nb,`
`int8_t x,`
`int8_t y) [inline]`

Moves sprite number **nb** relative to its current position.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	Number of pixels to move the sprite on the X axis Range: -128 - 127

Parameters

<i>y</i>	Number of pixels to move the sprite on the Y axis Range: -128 - 127
----------	---

See also

[move_sprite](#) for more details about the X and Y position

20.69.4.67 hide_sprite() `void hide_sprite (`
 `uint8_t nb) [inline]`

Hides sprite number **nb** by moving it to zero position by Y.

Parameters

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

20.69.4.68 set_vram_byte() `void set_vram_byte (`
 `uint8_t * addr,`
 `uint8_t v)`

Set byte in vram at given memory location

Parameters

<i>addr</i>	address to write to
<i>v</i>	value

20.69.4.69 set_attributed_tile_xy() `uint8_t* set_attributed_tile_xy (`
 `uint8_t x,`
 `uint8_t y,`
 `uint16_t t)`

Set single tile *t* with attributes on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

20.69.4.70 set_tile_xy() `uint8_t* set_tile_xy (`
 `uint8_t x,`
 `uint8_t y,`
 `uint8_t t)`

Set single tile *t* on background layer at x,y

Parameters

<i>x</i>	X-coordinate
<i>y</i>	Y-coordinate
<i>t</i>	tile index

Returns

returns the address of tile, so you may use faster [set_vram_byte\(\)](#) later

20.69.4.71 `get_bkg_xy_addr()` `uint8_t*` `get_bkg_xy_addr (`
 `uint8_t x,`
 `uint8_t y)`

Get address of X,Y tile of background map

20.69.5 Variable Documentation

20.69.5.1 `c` `void c`

20.69.5.2 `d` `void d`

20.69.5.3 `e` `void e`

20.69.5.4 `iyh` `void iyh`

20.69.5.5 `iyi` `uint8_t iyl`

20.69.5.6 `h` `uint8_t h`

20.69.5.7 `l` `void l`

20.69.5.8 `sys_time` `volatile uint16_t sys_time`

Global Time Counter in VBL periods (60Hz)

Increments once per Frame

Will wrap around every ~18 minutes (unsigned 16 bits = 65535 / 60 / 60 = 18.2)

20.69.5.9 `_current_2bpp_palette` `uint16_t _current_2bpp_palette`

20.69.5.10 `_current_1bpp_colors` `uint16_t _current_1bpp_colors`

20.69.5.11 `_map_tile_offset` `uint8_t _map_tile_offset`

20.69.5.12 `_submap_tile_offset` `uint8_t _submap_tile_offset`

20.69.5.13 `shadow_OAM` `volatile uint8_t shadow_OAM[]`

Shadow OAM array in WRAM, that is transferred into the real OAM each VBlank

20.69.5.14 `_shadow_OAM_base` `volatile uint8_t _shadow_OAM_base`

MSB of shadow_OAM address is used by OAM copying routine

MSB of shadow_OAM address is used by OAM DMA copying routine

20.69.5.15 `_shadow_OAM_OFF` `volatile uint8_t _shadow_OAM_OFF`

Flag for disabling of OAM copying routine

Values:

- 1: OAM copy routine is disabled (non-isr VDP operation may be in progress)
- 0: OAM copy routine is enabled

This flag is modified by all sms/gg GBDK API calls that write to the VDP. It is set to DISABLED when they start and ENABLED when they complete.

Note

It is recommended to avoid writing to the Video Display Processor (VDP) during an interrupt service routine (ISR) since it can corrupt the VDP pointer of an VDP operation already in progress.

If it is necessary, this flag can be used during an ISR to determine whether a VDP operation is already in progress.

If the value is 1 then avoid writing to the VDP (tiles, map, scrolling, colors, etc).

```
// at the beginning of and ISR that would write to the VDP
if (_shadow_OAM_OFF) return;
```

See also

[docs_consoles_safe_display_controller_access](#)

20.70 gbdk-lib/include/stdatomic.h File Reference

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

Data Structures

- struct [atomic_flag](#)

Functions

- `_Bool` [atomic_flag_test_and_set](#) (`volatile atomic_flag *object`) `OLDCALL`
- void [atomic_flag_clear](#) (`volatile atomic_flag *object`)

20.70.1 Function Documentation

20.70.1.1 `atomic_flag_test_and_set()` `_Bool atomic_flag_test_and_set (`
`volatile atomic_flag * object)`

20.70.1.2 `atomic_flag_clear()` `void atomic_flag_clear (`
`volatile atomic_flag * object)`

20.71 gbdk-lib/include/stdbool.h File Reference

Macros

- #define [true](#) ((_Bool)+1)
- #define [false](#) ((_Bool)+0)
- #define [bool](#) _Bool
- #define [__bool_true_false_are_defined](#) 1

20.71.1 Macro Definition Documentation

20.71.1.1 true #define true ((_Bool)+1)

20.71.1.2 false #define false ((_Bool)+0)

20.71.1.3 bool #define bool _Bool

20.71.1.4 __bool_true_false_are_defined #define __bool_true_false_are_defined 1

20.72 gbdk-lib/include/stddef.h File Reference

Macros

- #define [NULL](#) (void *)0
- #define [__PTRDIFF_T_DEFINED](#)
- #define [__SIZE_T_DEFINED](#)
- #define [__WCHAR_T_DEFINED](#)
- #define [offsetof](#)(s, m) __builtin_offsetof (s, m)

Typedefs

- typedef int [ptrdiff_t](#)
- typedef unsigned int [size_t](#)
- typedef unsigned long int [wchar_t](#)

20.72.1 Macro Definition Documentation

20.72.1.1 NULL #define NULL (void *)0

20.72.1.2 __PTRDIFF_T_DEFINED #define __PTRDIFF_T_DEFINED

20.72.1.3 __SIZE_T_DEFINED #define __SIZE_T_DEFINED

20.72.1.4 __WCHAR_T_DEFINED #define __WCHAR_T_DEFINED

20.72.1.5 offsetof `#define offsetof(
 s,
 m) __builtin_offsetof (s, m)`

20.72.2 Typedef Documentation

20.72.2.1 ptrdiff_t `typedef int ptrdiff_t`

20.72.2.2 size_t `typedef unsigned int size_t`

20.72.2.3 wchar_t `typedef unsigned long int wchar_t`

20.73 gbdk-lib/include/stdint.h File Reference

Macros

- `#define INT8_MIN (-128)`
- `#define INT16_MIN (-32767-1)`
- `#define INT32_MIN (-2147483647L-1)`
- `#define INT8_MAX (127)`
- `#define INT16_MAX (32767)`
- `#define INT32_MAX (2147483647L)`
- `#define UINT8_MAX (255)`
- `#define UINT16_MAX (65535)`
- `#define UINT32_MAX (4294967295UL)`
- `#define INT_LEAST8_MIN INT8_MIN`
- `#define INT_LEAST16_MIN INT16_MIN`
- `#define INT_LEAST32_MIN INT32_MIN`
- `#define INT_LEAST8_MAX INT8_MAX`
- `#define INT_LEAST16_MAX INT16_MAX`
- `#define INT_LEAST32_MAX INT32_MAX`
- `#define UINT_LEAST8_MAX UINT8_MAX`
- `#define UINT_LEAST16_MAX UINT16_MAX`
- `#define UINT_LEAST32_MAX UINT32_MAX`
- `#define INT_FAST8_MIN INT8_MIN`
- `#define INT_FAST16_MIN INT16_MIN`
- `#define INT_FAST32_MIN INT32_MIN`
- `#define INT_FAST8_MAX INT8_MAX`
- `#define INT_FAST16_MAX INT16_MAX`
- `#define INT_FAST32_MAX INT32_MAX`
- `#define UINT_FAST8_MAX UINT8_MAX`
- `#define UINT_FAST16_MAX UINT16_MAX`
- `#define UINT_FAST32_MAX UINT32_MAX`
- `#define INTPTR_MIN (-32767-1)`
- `#define INTPTR_MAX (32767)`
- `#define UINTPTR_MAX (65535)`
- `#define INTMAX_MIN (-2147483647L-1)`
- `#define INTMAX_MAX (2147483647L)`
- `#define UINTMAX_MAX (4294967295UL)`
- `#define PTRDIFF_MIN (-32767-1)`
- `#define PTRDIFF_MAX (32767)`
- `#define SIG_ATOMIC_MIN (0)`

- `#define SIG_ATOMIC_MAX (255)`
- `#define SIZE_MAX (65535u)`
- `#define INT8_C(c) c`
- `#define INT16_C(c) c`
- `#define INT32_C(c) c ## L`
- `#define UINT8_C(c) c ## U`
- `#define UINT16_C(c) c ## U`
- `#define UINT32_C(c) c ## UL`
- `#define WCHAR_MIN 0`
- `#define WCHAR_MAX 0xffffffff`
- `#define WINT_MIN 0`
- `#define WINT_MAX 0xffffffff`
- `#define INTMAX_C(c) c ## L`
- `#define UINTMAX_C(c) c ## UL`

Typedefs

- `typedef signed char int8_t`
- `typedef short int int16_t`
- `typedef long int int32_t`
- `typedef unsigned char uint8_t`
- `typedef unsigned short int uint16_t`
- `typedef unsigned long int uint32_t`
- `typedef signed char int_least8_t`
- `typedef short int int_least16_t`
- `typedef long int int_least32_t`
- `typedef unsigned char uint_least8_t`
- `typedef unsigned short int uint_least16_t`
- `typedef unsigned long int uint_least32_t`
- `typedef signed char int_fast8_t`
- `typedef int int_fast16_t`
- `typedef long int int_fast32_t`
- `typedef unsigned char uint_fast8_t`
- `typedef unsigned int uint_fast16_t`
- `typedef unsigned long int uint_fast32_t`
- `typedef int intptr_t`
- `typedef unsigned int uintptr_t`
- `typedef long int intmax_t`
- `typedef unsigned long int uintmax_t`

20.73.1 Macro Definition Documentation

20.73.1.1 INT8_MIN `#define INT8_MIN (-128)`

20.73.1.2 INT16_MIN `#define INT16_MIN (-32767-1)`

20.73.1.3 INT32_MIN `#define INT32_MIN (-2147483647L-1)`

20.73.1.4 INT8_MAX `#define INT8_MAX (127)`

20.73.1.5 INT16_MAX `#define INT16_MAX (32767)`

20.73.1.6 INT32_MAX `#define INT32_MAX (2147483647L)`

20.73.1.7 UINT8_MAX `#define UINT8_MAX (255)`

20.73.1.8 UINT16_MAX `#define UINT16_MAX (65535)`

20.73.1.9 UINT32_MAX `#define UINT32_MAX (4294967295UL)`

20.73.1.10 INT_LEAST8_MIN `#define INT_LEAST8_MIN INT8_MIN`

20.73.1.11 INT_LEAST16_MIN `#define INT_LEAST16_MIN INT16_MIN`

20.73.1.12 INT_LEAST32_MIN `#define INT_LEAST32_MIN INT32_MIN`

20.73.1.13 INT_LEAST8_MAX `#define INT_LEAST8_MAX INT8_MAX`

20.73.1.14 INT_LEAST16_MAX `#define INT_LEAST16_MAX INT16_MAX`

20.73.1.15 INT_LEAST32_MAX `#define INT_LEAST32_MAX INT32_MAX`

20.73.1.16 UINT_LEAST8_MAX `#define UINT_LEAST8_MAX UINT8_MAX`

20.73.1.17 UINT_LEAST16_MAX `#define UINT_LEAST16_MAX UINT16_MAX`

20.73.1.18 UINT_LEAST32_MAX `#define UINT_LEAST32_MAX UINT32_MAX`

20.73.1.19 INT_FAST8_MIN `#define INT_FAST8_MIN INT8_MIN`

20.73.1.20 INT_FAST16_MIN `#define INT_FAST16_MIN INT16_MIN`

20.73.1.21 INT_FAST32_MIN `#define INT_FAST32_MIN INT32_MIN`

20.73.1.22 INT_FAST8_MAX `#define INT_FAST8_MAX INT8_MAX`

20.73.1.23 INT_FAST16_MAX `#define INT_FAST16_MAX INT16_MAX`

20.73.1.24 INT_FAST32_MAX `#define INT_FAST32_MAX INT32_MAX`

20.73.1.25 UINT_FAST8_MAX `#define UINT_FAST8_MAX UINT8_MAX`

20.73.1.26 UINT_FAST16_MAX `#define UINT_FAST16_MAX UINT16_MAX`

20.73.1.27 UINT_FAST32_MAX `#define UINT_FAST32_MAX UINT32_MAX`

20.73.1.28 INTPTR_MIN `#define INTPTR_MIN (-32767-1)`

20.73.1.29 INTPTR_MAX `#define INTPTR_MAX (32767)`

20.73.1.30 UINTPTR_MAX `#define UINTPTR_MAX (65535)`

20.73.1.31 INTMAX_MIN `#define INTMAX_MIN (-2147483647L-1)`

20.73.1.32 INTMAX_MAX `#define INTMAX_MAX (2147483647L)`

20.73.1.33 UINTMAX_MAX `#define UINTMAX_MAX (4294967295UL)`

20.73.1.34 PTRDIFF_MIN `#define PTRDIFF_MIN (-32767-1)`

20.73.1.35 PTRDIFF_MAX `#define PTRDIFF_MAX (32767)`

20.73.1.36 SIG_ATOMIC_MIN `#define SIG_ATOMIC_MIN (0)`

20.73.1.37 SIG_ATOMIC_MAX `#define SIG_ATOMIC_MAX (255)`

20.73.1.38 SIZE_MAX `#define SIZE_MAX (65535u)`

20.73.1.39 INT8_C `#define INT8_C(
 c) c`

20.73.1.40 INT16_C `#define INT16_C(
 c) c`

20.73.1.41 INT32_C `#define INT32_C(
 c) c ## L`

20.73.1.42 UINT8_C `#define UINT8_C(
 c) c ## U`

20.73.1.43 UINT16_C `#define UINT16_C(
 c) c ## U`

20.73.1.44 UINT32_C `#define UINT32_C(
 c) c ## UL`

20.73.1.45 WCHAR_MIN `#define WCHAR_MIN 0`

20.73.1.46 WCHAR_MAX `#define WCHAR_MAX 0xffffffff`

20.73.1.47 WINT_MIN `#define WINT_MIN 0`

20.73.1.48 WINT_MAX `#define WINT_MAX 0xffffffff`

20.73.1.49 INTMAX_C `#define INTMAX_C(
 c) c ## L`

20.73.1.50 UINTMAX_C `#define UINTMAX_C(
 c) c ## UL`

20.73.2 Typedef Documentation

20.73.2.1 int8_t `typedef signed char int8_t`

20.73.2.2 int16_t `typedef short int int16_t`

20.73.2.3 int32_t `typedef long int int32_t`

20.73.2.4 uint8_t `typedef unsigned char uint8_t`

20.73.2.5 `uint16_t` typedef unsigned short int `uint16_t`

20.73.2.6 `uint32_t` typedef unsigned long int `uint32_t`

20.73.2.7 `int_least8_t` typedef signed char `int_least8_t`

20.73.2.8 `int_least16_t` typedef short int `int_least16_t`

20.73.2.9 `int_least32_t` typedef long int `int_least32_t`

20.73.2.10 `uint_least8_t` typedef unsigned char `uint_least8_t`

20.73.2.11 `uint_least16_t` typedef unsigned short int `uint_least16_t`

20.73.2.12 `uint_least32_t` typedef unsigned long int `uint_least32_t`

20.73.2.13 `int_fast8_t` typedef signed char `int_fast8_t`

20.73.2.14 `int_fast16_t` typedef int `int_fast16_t`

20.73.2.15 `int_fast32_t` typedef long int `int_fast32_t`

20.73.2.16 `uint_fast8_t` typedef unsigned char `uint_fast8_t`

20.73.2.17 `uint_fast16_t` typedef unsigned int `uint_fast16_t`

20.73.2.18 `uint_fast32_t` typedef unsigned long int `uint_fast32_t`

20.73.2.19 `intptr_t` typedef int `intptr_t`

20.73.2.20 `uintptr_t` typedef unsigned int `uintptr_t`

20.73.2.21 `intmax_t` typedef long int `intmax_t`

20.73.2.22 `uintmax_t` typedef unsigned long int `uintmax_t`

20.74 gbdk-lib/include/stdio.h File Reference

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

Functions

- void [putchar](#) (char *c*) [OLDCALL](#)
- void [printf](#) (const char **format*,...) [OLDCALL](#) [REENTRANT](#)
- void [sprintf](#) (char **str*, const char **format*,...) [OLDCALL](#) [REENTRANT](#)
- void [puts](#) (const char **s*)
- char * [gets](#) (char **s*) [OLDCALL](#)
- char [getchar](#) () [OLDCALL](#)

20.74.1 Detailed Description

Basic file/console input output functions.

Including stdio.h will use a large number of the background tiles for font characters. If stdio.h is not included then that space will be available for use with other tiles instead.

20.74.2 Function Documentation

20.74.2.1 putchar() `void putchar (`
 `char c)`

Print char to stdout.

Parameters

<i>c</i>	Character to print
----------	--------------------

20.74.2.2 printf() `void printf (`
 `const char * format,`
 `...)`

Print the string and arguments given by *format* to stdout.

Parameters

<i>format</i>	The format string as per printf
---------------	---------------------------------

Does not return the number of characters printed.

Currently supported:

- %hx (char as hex)
- %hu (unsigned char)
- %hd (signed char)
- %c (character)
- %u (unsigned int)
- %d (signed int)
- %x (unsigned int as hex)
- %s (string)

Warning: to correctly pass parameters (such as chars, ints, etc) **all of them should always be explicitly cast** as when calling the function. See [docs_chars_varargs](#) for more details.

20.74.2.3 sprintf() `void sprintf (`
`char * str,`
`const char * format,`
`...)`

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

Parameters

<i>str</i>	The buffer to print into
<i>format</i>	The format string as per printf

Does not return the number of characters printed.

Warning: to correctly pass parameters (such as chars, ints, etc) **all of them should always be explicitly cast** as when calling the function. See [docs_chars_varargs](#) for more details.

20.74.2.4 puts() `void puts (`
`const char * s)`

[puts\(\)](#) writes the string **s** and a trailing newline to stdout.

20.74.2.5 gets() `char* gets (`
`char * s)`

[gets\(\)](#) Reads a line from stdin into a buffer pointed to by **s**.

Parameters

<i>s</i>	Buffer to store string in
----------	---------------------------

Reads until either a terminating newline or an EOF, which it replaces with '\0'. No check for buffer overrun is performed.

Returns: Buffer pointed to by **s**

20.74.2.6 getchar() `char getchar ()`

[getchar\(\)](#) Reads and returns a single character from stdin.

20.75 gbdk-lib/include/stdlib.h File Reference

`#include <types.h>`

Macros

- `#define __reentrant`

Functions

- void [exit](#) (int status) [OLDCALL](#)
- int [abs](#) (int i) [OLDCALL](#)
- long [labs](#) (long num) [OLDCALL](#)
- int [atoi](#) (const char *s)
- long [atol](#) (const char *s)
- char * [itoa](#) (int n, char *s, unsigned char radix) [OLDCALL](#)
- char * [uitoa](#) (unsigned int n, char *s, unsigned char radix) [OLDCALL](#)
- char * [ltoa](#) (long n, char *s, unsigned char radix) [OLDCALL](#)

- char * [ultoa](#) (unsigned long n, char *s, unsigned char radix) [OLDCALL](#)
- void * [calloc](#) (size_t nmemb, size_t size)
- void * [malloc](#) (size_t size)
- void * [realloc](#) (void *ptr, size_t size)
- void [free](#) (void *ptr)
- void * [bsearch](#) (const void *key, const void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *)) [__reentrant](#))
- void [qsort](#) (void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *)) [__reentrant](#))

20.75.1 Macro Definition Documentation

20.75.1.1 [__reentrant](#) `#define __reentrant`
file stdlib.h 'Standard library' functions, for whatever that means.

20.75.2 Function Documentation

20.75.2.1 [exit\(\)](#) `void exit (`
 `int status)`

Causes normal program termination and the value of status is returned to the parent. All open streams are flushed and closed.

20.75.2.2 [abs\(\)](#) `int abs (`
 `int i)`

Returns the absolute value of int i

Parameters

<i>i</i>	Int to obtain absolute value of
----------	---------------------------------

If i is negative, returns -i; else returns i.

20.75.2.3 [labs\(\)](#) `long labs (`
 `long num)`

Returns the absolute value of long int num

Parameters

<i>num</i>	Long integer to obtain absolute value of
------------	--

20.75.2.4 [atoi\(\)](#) `int atoi (`
 `const char * s)`

Converts an ASCII string to an int

Parameters

<i>s</i>	String to convert to an int
----------	-----------------------------

The string may be of the format

`[\s]*[+-][\d]+[\D]*`

i.e. any number of spaces, an optional + or -, then an arbitrary number of digits.

The result is undefined if the number doesn't fit in an int.
Returns: Int value of string

20.75.2.5 atol() `long atol (`
 `const char * s)`

Converts an ASCII string to a long.

Parameters

<i>s</i>	String to convert to a long int
----------	---------------------------------

See also

[atoi\(\)](#)

Returns: Long int value of string

20.75.2.6 itoa() `char* itoa (`
 `int n,`
 `char * s,`
 `unsigned char radix)`

Converts an int into a base 10 ASCII string.

Parameters

<i>n</i>	Int to convert to a string
<i>s</i>	String to store the converted number
<i>radix</i>	Numerical base for converted number, ex: 10 is decimal base (parameter is required but not utilized on Game Boy and Analogue Pocket)

Can be used with [set_bkg_based_tiles\(\)](#) for printing if the digit character tiles are not ascii-mapped.
Returns: Pointer to converted string

20.75.2.7 uitoa() `char* uitoa (`
 `unsigned int n,`
 `char * s,`
 `unsigned char radix)`

Converts an unsigned int into a base 10 ASCII string.

Parameters

<i>n</i>	Unsigned Int to convert to a string
<i>s</i>	String to store the converted number
<i>radix</i>	Numerical base for converted number, ex: 10 is decimal base (parameter is required but not utilized on Game Boy and Analogue Pocket)

Can be used with [set_bkg_based_tiles\(\)](#) for printing if the digit character tiles are not ascii-mapped.
Returns: Pointer to converted string

20.75.2.8 ltoa() `char* ltoa (`
 `long n,`
 `char * s,`
 `unsigned char radix)`

Converts a long into a base 10 ASCII string.

Parameters

<i>n</i>	Long int to convert to a string
<i>s</i>	String to store the converted number
<i>radix</i>	Numerical base for converted number, ex: 10 is decimal base (parameter is required but not utilized on Game Boy and Analogue Pocket)

Can be used with [set_bkg_based_tiles\(\)](#) for printing if the digit character tiles are not ascii-mapped.

Returns: Pointer to converted string

20.75.2.9 ultoa() `char* ultoa (`
 `unsigned long n,`
 `char * s,`
 `unsigned char radix)`

Converts an unsigned long into a base 10 ASCII string.

Parameters

<i>n</i>	Unsigned Long Int to convert to a string
<i>s</i>	String to store the converted number
<i>radix</i>	Numerical base for converted number, ex: 10 is decimal base (parameter is required but not utilized on Game Boy and Analogue Pocket)

Can be used with [set_bkg_based_tiles\(\)](#) for printing if the digit character tiles are not ascii-mapped.

Returns: Pointer to converted string

20.75.2.10 calloc() `void* calloc (`
 `size_t nmemb,`
 `size_t size)`

Memory allocation functions

20.75.2.11 malloc() `void* malloc (`
 `size_t size)`

20.75.2.12 realloc() `void* realloc (`
 `void * ptr,`
 `size_t size)`

20.75.2.13 free() `void free (`
 `void * ptr)`

20.75.2.14 bsearch() `void* bsearch (`
 `const void * key,`
 `const void * base,`
 `size_t nmemb,`
 `size_t size,`
 `int (*)(const void *, const void *) __reentrant compar)`

search a sorted array of **nmemb** items

Parameters

<i>key</i>	Pointer to object that is the key for the search
------------	--

Parameters

<i>base</i>	Pointer to first object in the array to search
<i>nmemb</i>	Number of elements in the array
<i>size</i>	Size in bytes of each element in the array
<i>compar</i>	Function used to compare two elements of the array

Returns: Pointer to array entry that matches the search key. If key is not found, NULL is returned.

20.75.2.15 qsort() `void qsort (`
`void * base,`
`size_t nmemb,`
`size_t size,`
`int (*) (const void *, const void *) __reentrant compar)`

Sort an array of **nmemb** items

Parameters

<i>base</i>	Pointer to first object in the array to sort
<i>nmemb</i>	Number of elements in the array
<i>size</i>	Size in bytes of each element in the array
<i>compar</i>	Function used to compare and sort two elements of the array

20.76 gbdk-lib/include/stdnoreturn.h File Reference

Macros

- #define `noreturn` `_Noreturn`

20.76.1 Macro Definition Documentation

20.76.1.1 noreturn `#define noreturn _Noreturn`

20.77 gbdk-lib/include/time.h File Reference

```
#include <types.h>
#include <stdint.h>
```

Macros

- #define `CLOCKS_PER_SEC` 60

Typedefs

- typedef `uint16_t` `time_t`

Functions

- `clock_t` `clock ()` `OLDCALL`
- `time_t` `time (time_t *t)`

20.77.1 Detailed Description

Sort of ANSI compliant time functions.

20.77.2 Macro Definition Documentation

20.77.2.1 CLOCKS_PER_SEC `#define CLOCKS_PER_SEC 60`

20.77.3 Typedef Documentation

20.77.3.1 time_t `typedef uint16_t time_t`

20.77.4 Function Documentation

20.77.4.1 clock() `clock_t clock ()`

Returns an approximation of processor time used by the program in Clocks

The value returned is the CPU time (ticks) used so far as a `clock_t`.

To get the number of seconds used, divide by `CLOCKS_PER_SEC`.

This is based on `sys_time`, which will wrap around every ~18 minutes. (unsigned 16 bits = 65535 / 60 / 60 = 18.2)

See also

`sys_time`, `time()`

20.77.4.2 time() `time_t time (`
`time_t * t)`

Converts `clock()` time to Seconds

Parameters

<code>t</code>	If pointer <code>t</code> is not NULL, it's value will be set to the same seconds calculation as returned by the function.
----------------	--

The calculation is `clock()` / `CLOCKS_PER_SEC`

Returns: time in seconds

See also

`sys_time`, `clock()`

20.78 gbdk-lib/include/typeof.h File Reference

Macros

- `#define TYPEOF_INT 1`
- `#define TYPEOF_SHORT 2`
- `#define TYPEOF_CHAR 3`
- `#define TYPEOF_LONG 4`
- `#define TYPEOF_FLOAT 5`
- `#define TYPEOF_FIXED16X16 6`
- `#define TYPEOF_BIT 7`
- `#define TYPEOF_BITFIELD 8`
- `#define TYPEOF_SBIT 9`

- `#define` [TYPEOF_SFR](#) 10
- `#define` [TYPEOF_VOID](#) 11
- `#define` [TYPEOF_STRUCT](#) 12
- `#define` [TYPEOF_ARRAY](#) 13
- `#define` [TYPEOF_FUNCTION](#) 14
- `#define` [TYPEOF_POINTER](#) 15
- `#define` [TYPEOF_FPOINTER](#) 16
- `#define` [TYPEOF_CPOINTER](#) 17
- `#define` [TYPEOF_GPOINTER](#) 18
- `#define` [TYPEOF_PPOINTER](#) 19
- `#define` [TYPEOF_IPOINTER](#) 20
- `#define` [TYPEOF_EEPPPOINTER](#) 21

20.78.1 Macro Definition Documentation

20.78.1.1 TYPEOF_INT `#define TYPEOF_INT 1`

20.78.1.2 TYPEOF_SHORT `#define TYPEOF_SHORT 2`

20.78.1.3 TYPEOF_CHAR `#define TYPEOF_CHAR 3`

20.78.1.4 TYPEOF_LONG `#define TYPEOF_LONG 4`

20.78.1.5 TYPEOF_FLOAT `#define TYPEOF_FLOAT 5`

20.78.1.6 TYPEOF_FIXED16X16 `#define TYPEOF_FIXED16X16 6`

20.78.1.7 TYPEOF_BIT `#define TYPEOF_BIT 7`

20.78.1.8 TYPEOF_BITFIELD `#define TYPEOF_BITFIELD 8`

20.78.1.9 TYPEOF_SBIT `#define TYPEOF_SBIT 9`

20.78.1.10 TYPEOF_SFR `#define TYPEOF_SFR 10`

20.78.1.11 TYPEOF_VOID `#define TYPEOF_VOID 11`

20.78.1.12 TYPEOF_STRUCT `#define TYPEOF_STRUCT 12`

20.78.1.13 TYPEOF_ARRAY `#define TYPEOF_ARRAY 13`

20.78.1.14 **TYPEOF_FUNCTION** `#define TYPEOF_FUNCTION 14`

20.78.1.15 **TYPEOF_POINTER** `#define TYPEOF_POINTER 15`

20.78.1.16 **TYPEOF_FPOINTER** `#define TYPEOF_FPOINTER 16`

20.78.1.17 **TYPEOF_CPOINTER** `#define TYPEOF_CPOINTER 17`

20.78.1.18 **TYPEOF_GPOINTER** `#define TYPEOF_GPOINTER 18`

20.78.1.19 **TYPEOF_PPOINTER** `#define TYPEOF_PPOINTER 19`

20.78.1.20 **TYPEOF_IPOINTER** `#define TYPEOF_IPOINTER 20`

20.78.1.21 **TYPEOF_EEPPPOINTER** `#define TYPEOF_EEPPPOINTER 21`

Index

- `_AUD3WAVERAM`
 - `hardware.h`, [176](#)
- `_BIOS`
 - `hardware.h`, [188](#), [201](#)
- `_HRAM`
 - `hardware.h`, [176](#)
- `_IO`
 - `hardware.h`, [176](#)
- `_OAMRAM`
 - `hardware.h`, [175](#)
- `_RAM`
 - `hardware.h`, [175](#)
- `_RAMBANK`
 - `hardware.h`, [175](#)
- `_SCRN0`
 - `hardware.h`, [175](#)
- `_SCRN1`
 - `hardware.h`, [175](#)
- `_SRAM`
 - `hardware.h`, [175](#)
- `_SYSTEM`
 - `hardware.h`, [189](#), [201](#)
- `_VRAM`
 - `hardware.h`, [175](#)
- `_VRAM8000`
 - `hardware.h`, [175](#)
- `_VRAM8800`
 - `hardware.h`, [175](#)
- `_VRAM9000`
 - `hardware.h`, [175](#)
- `__BYTES`
 - `hardware.h`, [162](#), [182](#), [190](#), [195](#)
- `__BYTE_REG`
 - `hardware.h`, [162](#), [182](#), [190](#), [195](#)
- `__GBDK_VERSION`
 - `version.h`, [232](#)
- `__HandleCrash`
 - `crash_handler.h`, [102](#)
- `__PTRDIFF_T_DEFINED`
 - `stddef.h`, [311](#)
- `__READ_VDP_REG`
 - `msx.h`, [238](#)
 - `sms.h`, [292](#)
- `__REG`
 - `hardware.h`, [162](#)
- `__SIZE_T_DEFINED`
 - `stddef.h`, [311](#)
 - `types.h`, [87](#), [88](#), [91](#)
- `__WCHAR_T_DEFINED`
 - `stddef.h`, [311](#)
- `__WRITE_VDP_REG`
 - `msx.h`, [238](#)
 - `sms.h`, [292](#)
- `__assert`
 - `assert.h`, [92](#)
- `__bool_true_false_are_defined`
 - `stdbool.h`, [311](#)
- `__call_banked`
 - `far_ptr.h`, [226](#)
- `__call_banked_addr`
 - `far_ptr.h`, [226](#)
- `__call_banked_bank`
 - `far_ptr.h`, [226](#)
- `__call_banked_ptr`
 - `far_ptr.h`, [226](#)
- `__current_base_tile`
 - `metasprites.h`, [209](#), [211](#), [216](#), [220](#)
- `__current metasprite`
 - `metasprites.h`, [208](#), [211](#), [216](#), [219](#)
- `__far_ptr`, [64](#)
 - `fn`, [65](#)
 - `ofs`, [65](#)
 - `ptr`, [65](#)
 - `seg`, [65](#)
 - `segin`, [65](#)
 - `segofs`, [65](#)
- `__rand_seed`
 - `rand.h`, [286](#)
- `__reentrant`
 - `stdlib.h`, [320](#)
- `__render_shadow_OAM`
 - `metasprites.h`, [209](#), [211](#), [216](#), [220](#)
- `__setjmp`
 - `setjmp.h`, [287](#)
- `__cpu`
 - `gb.h`, [152](#)
- `__current_1bpp_colors`
 - `gb.h`, [153](#)
 - `msx.h`, [255](#)
 - `nes.h`, [284](#)
 - `sms.h`, [309](#)
- `__current_2bpp_palette`
 - `msx.h`, [255](#)
 - `sms.h`, [309](#)
- `__current_bank`
 - `gb.h`, [152](#)
 - `msx.h`, [255](#)
 - `nes.h`, [284](#)
 - `sms.h`, [295](#)
- `__fixed`, [65](#)
 - `b`, [66](#)
 - `h`, [65](#)
 - `l`, [65](#)
 - `w`, [66](#)
- `__io_in`
 - `gb.h`, [152](#)
- `__io_out`
 - `gb.h`, [152](#)
- `__io_status`
 - `gb.h`, [152](#)

- `_is_GBA`
 - `gb.h`, 152
- `_map_tile_offset`
 - `gb.h`, 153
 - `msx.h`, 255
 - `nes.h`, 284
 - `sms.h`, 309
- `_shadow_OAM_OFF`
 - `msx.h`, 255
 - `sms.h`, 310
- `_shadow_OAM_base`
 - `gb.h`, 153
 - `msx.h`, 255
 - `nes.h`, 284
 - `sms.h`, 310
- `_submap_tile_offset`
 - `gb.h`, 153
 - `msx.h`, 255
 - `nes.h`, 284
 - `sms.h`, 309
- `abs`
 - `stdlib.h`, 320
- `add_JOY`
 - `gb.h`, 127
 - `msx.h`, 245
 - `sms.h`, 299
- `add_LCD`
 - `gb.h`, 126
 - `msx.h`, 245
 - `sms.h`, 299
- `add_low_priority_TIM`
 - `gb.h`, 126
- `add_SIO`
 - `gb.h`, 126
 - `msx.h`, 245
 - `sms.h`, 299
- `add_TIM`
 - `gb.h`, 126
 - `msx.h`, 245
 - `sms.h`, 299
- `add_VBL`
 - `gb.h`, 125
 - `msx.h`, 245
 - `sms.h`, 299
- `AND`
 - `drawing.h`, 104
- `arand`
 - `rand.h`, 286
- `assert`
 - `assert.h`, 92
- `assert.h`
 - `__assert`, 92
 - `assert`, 92
- `AT`
 - `types.h`, 89
- `atoi`
 - `stdlib.h`, 320
- `atol`
 - `stdlib.h`, 321
- `atomic_flag`, 66
 - `flag`, 66
- `atomic_flag_clear`
 - `stdatomic.h`, 310
- `atomic_flag_test_and_set`
 - `stdatomic.h`, 310
- `AUD1SWEEP_DOWN`
 - `hardware.h`, 164
- `AUD1SWEEP_LENGTH`
 - `hardware.h`, 165
- `AUD1SWEEP_TIME`
 - `hardware.h`, 164
- `AUD1SWEEP_UP`
 - `hardware.h`, 164
- `AUD3WAVE`
 - `hardware.h`, 178
- `AUD4POLY_WIDTH_15BIT`
 - `hardware.h`, 166
- `AUD4POLY_WIDTH_7BIT`
 - `hardware.h`, 166
- `AUDENA_OFF`
 - `hardware.h`, 167
- `AUDENA_ON`
 - `hardware.h`, 167
- `AUDENV_DOWN`
 - `hardware.h`, 173
- `AUDENV_LENGTH`
 - `hardware.h`, 173
- `AUDENV_UP`
 - `hardware.h`, 173
- `AUDENV_VOL`
 - `hardware.h`, 173
- `AUDHIGH_LENGTH_OFF`
 - `hardware.h`, 173
- `AUDHIGH_LENGTH_ON`
 - `hardware.h`, 173
- `AUDHIGH_RESTART`
 - `hardware.h`, 173
- `AUDLEN_DUTY_12_5`
 - `hardware.h`, 173
- `AUDLEN_DUTY_25`
 - `hardware.h`, 173
- `AUDLEN_DUTY_50`
 - `hardware.h`, 173
- `AUDLEN_DUTY_75`
 - `hardware.h`, 173
- `AUDLEN_LENGTH`
 - `hardware.h`, 173
- `AUDTERM_1_LEFT`
 - `hardware.h`, 166
- `AUDTERM_1_RIGHT`
 - `hardware.h`, 166
- `AUDTERM_2_LEFT`
 - `hardware.h`, 166
- `AUDTERM_2_RIGHT`
 - `hardware.h`, 166
- `AUDTERM_3_LEFT`
 - `hardware.h`, 166

- hardware.h, [166](#)
- AUDTERM_3_RIGHT
 - hardware.h, [166](#)
- AUDTERM_4_LEFT
 - hardware.h, [166](#)
- AUDTERM_4_RIGHT
 - hardware.h, [166](#)
- AUDVOL_VIN_LEFT
 - hardware.h, [166](#)
- AUDVOL_VIN_RIGHT
 - hardware.h, [166](#)
- AUDVOL_VOL_LEFT
 - hardware.h, [166](#)
- AUDVOL_VOL_RIGHT
 - hardware.h, [166](#)
- b
 - _fixed, [66](#)
 - gb.h, [152](#)
- BANK
 - gb.h, [119](#)
 - incbin.h, [229](#)
 - msx.h, [241](#)
 - nes.h, [262](#)
 - sms.h, [295](#)
- BANKED
 - types.h, [89](#)
- BANKREF
 - gb.h, [119](#)
 - msx.h, [241](#)
 - nes.h, [264](#)
 - sms.h, [295](#)
- BANKREF_EXTERN
 - gb.h, [119](#)
 - msx.h, [241](#)
 - nes.h, [264](#)
 - sms.h, [295](#)
- BCD
 - bcd.h, [95](#)
- bcd.h
 - BCD, [95](#)
 - bcd2text, [96](#)
 - bcd_add, [95](#)
 - BCD_HEX, [95](#)
 - bcd_sub, [95](#)
 - MAKE_BCD, [95](#)
 - uint2bcd, [95](#)
- bcd2text
 - bcd.h, [96](#)
- bcd_add
 - bcd.h, [95](#)
- BCD_HEX
 - bcd.h, [95](#)
- bcd_sub
 - bcd.h, [95](#)
- BCPD_REG
 - hardware.h, [179](#)
- BCPS_REG
 - hardware.h, [179](#)
- BCPSF_AUTOINC
 - hardware.h, [172](#)
- BGB_BREAKPOINT
 - emu_debug.h, [109](#)
- BGB_MESSAGE
 - emu_debug.h, [108](#)
- BGB_printf
 - emu_debug.h, [109](#)
- BGB_PROFILE_BEGIN
 - emu_debug.h, [108](#)
- BGB_TEXT
 - emu_debug.h, [109](#)
- BGP_REG
 - hardware.h, [178](#)
- bkg_scroll_x
 - hardware.h, [192](#)
- bkg_scroll_y
 - hardware.h, [192](#)
- BKGF_BANK0
 - hardware.h, [170](#)
- BKGF_BANK1
 - hardware.h, [170](#)
- BKGF_CGB_PAL0
 - hardware.h, [170](#)
- BKGF_CGB_PAL1
 - hardware.h, [171](#)
- BKGF_CGB_PAL2
 - hardware.h, [171](#)
- BKGF_CGB_PAL3
 - hardware.h, [171](#)
- BKGF_CGB_PAL4
 - hardware.h, [171](#)
- BKGF_CGB_PAL5
 - hardware.h, [171](#)
- BKGF_CGB_PAL6
 - hardware.h, [171](#)
- BKGF_CGB_PAL7
 - hardware.h, [171](#)
- BKGF_PRI
 - hardware.h, [170](#)
- BKGF_XFLIP
 - hardware.h, [170](#)
- BKGF_YFLIP
 - hardware.h, [170](#)
- BLACK
 - drawing.h, [104](#)
- bool
 - stdbool.h, [311](#)
- BOOLEAN
 - types.h, [89](#)
- box
 - drawing.h, [106](#)
- BP_SIZE
 - setjmp.h, [287](#)
- BPX_SIZE
 - setjmp.h, [287](#)
- bsearch
 - stdlib.h, [322](#)

BYTE

types.h, 89

c

gb.h, 151

gbdecompress.h, 156, 157

metasprites.h, 209

msx.h, 254

sgb.h, 223

sms.h, 309

string.h, 82

calloc

stdlib.h, 322

cancel_pending_interrupts

gb.h, 127

msx.h, 245

sms.h, 299

cgb.h

cgb_compatibility, 102

cpu_fast, 101

cpu_slow, 101

palette_color_t, 99

RGB, 97

RGB8, 98

RGB_AQUA, 99

RGB_BLACK, 99

RGB_BLUE, 98

RGB_BROWN, 99

RGB_CYAN, 99

RGB_DARKBLUE, 99

RGB_DARKGRAY, 99

RGB_DARKGREEN, 98

RGB_DARKRED, 98

RGB_DARKYELLOW, 99

RGB_GREEN, 98

RGB_LIGHTFLESH, 99

RGB_LIGHTGRAY, 99

RGB_ORANGE, 99

RGB_PINK, 99

RGB_PURPLE, 99

RGB_RED, 98

RGB_TEAL, 99

RGB_WHITE, 99

RGB_YELLOW, 99

RGBHTML, 98

set_bkg_palette, 100

set_bkg_palette_entry, 100

set_default_palette, 102

set_sprite_palette, 100

set_sprite_palette_entry, 101

cgb_compatibility

cgb.h, 102

sms.h, 301

CGB_TYPE

gb.h, 118

CHAR_BIT

limits.h, 232

CHAR_MAX

limits.h, 232

CHAR_MIN

limits.h, 232

circle

drawing.h, 106

clock

time.h, 324

clock_t

types.h, 87, 88, 91

CLOCKS_PER_SEC

time.h, 324

cls

console.h, 224

color

drawing.h, 106

COMPAT_PALETTE

gb.h, 123

msx.h, 242

nes.h, 266

sms.h, 297

console.h

cls, 224

gotoxy, 223

posx, 223

posy, 223

setchar, 223

cpu_fast

cgb.h, 101

msx.h, 247

sms.h, 301

cpu_slow

cgb.h, 101

crash_handler.h

__HandleCrash, 102

CRITICAL

types.h, 89

ctype.h

isalpha, 93

isdigit, 93

islower, 93

isspace, 93

isupper, 93

tolower, 94

toupper, 94

CURRENT_BANK

gb.h, 119

msx.h, 240

nes.h, 262

sms.h, 295

d

gb.h, 152

msx.h, 254

sms.h, 309

delay

gb.h, 128

msx.h, 246

nes.h, 267

sms.h, 300

DEVICE_SCREEN_BUFFER_HEIGHT

- hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_BUFFER_WIDTH
 - hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_HEIGHT
 - hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_MAP_ENTRY_SIZE
 - hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_PX_HEIGHT
 - hardware.h, [175](#), [188](#), [192](#), [201](#)
- DEVICE_SCREEN_PX_WIDTH
 - hardware.h, [175](#), [188](#), [192](#), [200](#)
- DEVICE_SCREEN_WIDTH
 - hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_X_OFFSET
 - hardware.h, [174](#), [191](#)
- DEVICE_SCREEN_Y_OFFSET
 - hardware.h, [174](#), [191](#)
- DEVICE_SPRITE_PX_OFFSET_X
 - hardware.h, [175](#), [192](#)
- DEVICE_SPRITE_PX_OFFSET_Y
 - hardware.h, [175](#), [192](#)
- DEVICE_SUPPORTS_COLOR
 - gb.h, [118](#)
 - msx.h, [240](#)
 - sms.h, [295](#)
- DEVICE_WINDOW_PX_OFFSET_X
 - hardware.h, [175](#), [192](#)
- DEVICE_WINDOW_PX_OFFSET_Y
 - hardware.h, [175](#), [192](#)
- disable_interrupts
 - gb.h, [130](#)
 - nes.h, [269](#)
- DISABLE_OAM_DMA
 - gb.h, [124](#)
 - nes.h, [266](#)
- DISABLE_RAM
 - gb.h, [121](#)
 - msx.h, [242](#)
 - sms.h, [296](#)
- DISABLE_RAM_MBC1
 - gb.h, [121](#)
- DISABLE_RAM_MBC5
 - gb.h, [122](#)
- DISABLE_VBL_TRANSFER
 - gb.h, [124](#)
 - msx.h, [243](#)
 - nes.h, [266](#)
 - sms.h, [297](#)
- DISPLAY_OFF
 - gb.h, [122](#)
 - msx.h, [239](#)
 - nes.h, [265](#)
 - sms.h, [294](#)
- display_off
 - gb.h, [130](#)
 - msx.h, [245](#)
 - nes.h, [270](#)
 - sms.h, [300](#)
- DISPLAY_ON
 - gb.h, [122](#)
 - msx.h, [239](#)
 - nes.h, [265](#)
 - sms.h, [293](#)
- DIV_REG
 - hardware.h, [176](#)
- DKGREY
 - drawing.h, [104](#)
- DMA_REG
 - hardware.h, [178](#)
- DMG_BLACK
 - gb.h, [116](#)
 - nes.h, [262](#)
- DMG_DARK_GRAY
 - gb.h, [116](#)
 - nes.h, [262](#)
- DMG_LITE_GRAY
 - gb.h, [116](#)
 - nes.h, [262](#)
- DMG_PALETTE
 - gb.h, [117](#)
 - nes.h, [262](#)
- DMG_TYPE
 - gb.h, [118](#)
- DMG_WHITE
 - gb.h, [116](#)
 - nes.h, [262](#)
- docs/pages/01_getting_started.md, [72](#)
- docs/pages/02_links_and_tools.md, [72](#)
- docs/pages/03_using_gbdk.md, [72](#)
- docs/pages/04_coding_guidelines.md, [72](#)
- docs/pages/05_banking_mbc5.md, [72](#)
- docs/pages/06_toolchain.md, [72](#)
- docs/pages/06b_supported_consoles.md, [72](#)
- docs/pages/07_sample_programs.md, [72](#)
- docs/pages/08_faq.md, [72](#)
- docs/pages/09_migrating_new_versions.md, [72](#)
- docs/pages/10_release_notes.md, [72](#)
- docs/pages/20_toolchain_settings.md, [72](#)
- docs/pages/docs_index.md, [72](#)
- draw_image
 - drawing.h, [106](#)
- drawing.h
 - AND, [104](#)
 - BLACK, [104](#)
 - box, [106](#)
 - circle, [106](#)
 - color, [106](#)
 - DKGREY, [104](#)
 - draw_image, [106](#)
 - getpix, [106](#)
 - gotogxy, [106](#)
 - gprint, [104](#)
 - gprintf, [105](#)
 - gprintln, [104](#)
 - gprintn, [105](#)
 - GRAPHICS_HEIGHT, [103](#)

GRAPHICS_WIDTH, [103](#)
 line, [106](#)
 LTGREY, [104](#)
 M_FILL, [104](#)
 M_NOFILL, [104](#)
 OR, [103](#)
 plot, [105](#)
 plot_point, [105](#)
 SIGNED, [104](#)
 SOLID, [103](#)
 switch_data, [105](#)
 UNSIGNED, [104](#)
 WHITE, [104](#)
 wrtchr, [106](#)
 XOR, [103](#)

dtile

- metasprite_t, [70](#)

DWORD

- types.h, [90](#)

dx

- metasprite_t, [70](#)

dy

- metasprite_t, [70](#)

e

- gb.h, [153](#)
- msx.h, [254](#)
- sms.h, [309](#)

EMPTY_IFLAG

- gb.h, [115](#)
- msx.h, [238](#)
- sms.h, [292](#)

EMU_BREAKPOINT

- emu_debug.h, [109](#)

emu_debug.h

- BGB_BREAKPOINT, [109](#)
- BGB_MESSAGE, [108](#)
- BGB_printf, [109](#)
- BGB_PROFILE_BEGIN, [108](#)
- BGB_TEXT, [109](#)
- EMU_BREAKPOINT, [109](#)
- EMU_MESSAGE, [107](#)
- EMU_printf, [109](#)
- EMU_PROFILE_BEGIN, [108](#)
- EMU_TEXT, [108](#)

EMU_MESSAGE

- emu_debug.h, [107](#)

EMU_printf

- emu_debug.h, [109](#)

EMU_PROFILE_BEGIN

- emu_debug.h, [108](#)

EMU_TEXT

- emu_debug.h, [108](#)

enable_interrupts

- gb.h, [129](#)
- nes.h, [269](#)

ENABLE_OAM_DMA

- gb.h, [124](#)
- nes.h, [266](#)

ENABLE_RAM

- gb.h, [121](#)
- msx.h, [242](#)
- sms.h, [296](#)

ENABLE_RAM_MBC1

- gb.h, [121](#)

ENABLE_RAM_MBC5

- gb.h, [122](#)

ENABLE_VBL_TRANSFER

- gb.h, [124](#)
- msx.h, [243](#)
- nes.h, [266](#)
- sms.h, [297](#)

exit

- stdlib.h, [320](#)

FALSE

- types.h, [92](#)

false

- stdbool.h, [311](#)

FAR_CALL

- far_ptr.h, [225](#)

FAR_FUNC

- far_ptr.h, [225](#)

FAR_OFS

- far_ptr.h, [225](#)

FAR_PTR

- far_ptr.h, [226](#)

far_ptr.h

- __call_banked, [226](#)
- __call_banked_addr, [226](#)
- __call_banked_bank, [226](#)
- __call_banked_ptr, [226](#)
- FAR_CALL, [225](#)
- FAR_FUNC, [225](#)
- FAR_OFS, [225](#)
- FAR_PTR, [226](#)
- FAR_SEG, [225](#)
- TO_FAR_PTR, [224](#)
- to_far_ptr, [226](#)

FAR_SEG

- far_ptr.h, [225](#)

fill_bkg_rect

- gb.h, [151](#)
- msx.h, [243](#)
- nes.h, [283](#)
- sms.h, [297](#)

fill_rect

- gb.h, [124](#)
- msx.h, [251](#)
- nes.h, [266](#)
- sms.h, [306](#)

fill_rect_compat

- sms.h, [306](#)

fill_win_rect

- gb.h, [151](#)
- msx.h, [243](#)
- sms.h, [297](#)

first_tile

- sfont_handle, [71](#)
- fixed
 - types.h, [90](#)
- flag
 - atomic_flag, [66](#)
- fn
 - __far_ptr, [65](#)
- font
 - sfont_handle, [71](#)
- font.h
 - FONT_128ENCODING, [227](#)
 - FONT_256ENCODING, [227](#)
 - font_color, [228](#)
 - FONT_COMPRESSED, [227](#)
 - font_init, [228](#)
 - font_load, [228](#)
 - FONT_NOENCODING, [227](#)
 - font_set, [228](#)
 - font_t, [228](#)
 - mfont_handle, [228](#)
 - pmfont_handle, [228](#)
- FONT_128ENCODING
 - font.h, [227](#)
- FONT_256ENCODING
 - font.h, [227](#)
- font_color
 - font.h, [228](#)
- FONT_COMPRESSED
 - font.h, [227](#)
- font_ibm
 - List of gbdk fonts, [64](#)
- font_ibm_fixed
 - List of gbdk fonts, [64](#)
- font_init
 - font.h, [228](#)
- font_italic
 - List of gbdk fonts, [64](#)
- font_load
 - font.h, [228](#)
- font_min
 - List of gbdk fonts, [64](#)
- FONT_NOENCODING
 - font.h, [227](#)
- font_set
 - font.h, [228](#)
- font_spect
 - List of gbdk fonts, [64](#)
- font_t
 - font.h, [228](#)
- free
 - stdlib.h, [322](#)
- func
 - isr_nested_vector_t, [66](#)
 - isr_vector_t, [67](#)
- GAMEBOY
 - gb.h, [114](#)
- gb.h
 - _cpu, [152](#)
 - _current_1bpp_colors, [153](#)
 - _current_bank, [152](#)
 - _io_in, [152](#)
 - _io_out, [152](#)
 - _io_status, [152](#)
 - _is_GBA, [152](#)
 - _map_tile_offset, [153](#)
 - _shadow_OAM_base, [153](#)
 - _submap_tile_offset, [153](#)
 - add_JOY, [127](#)
 - add_LCD, [126](#)
 - add_low_priority_TIM, [126](#)
 - add_SIO, [126](#)
 - add_TIM, [126](#)
 - add_VBL, [125](#)
 - b, [152](#)
 - BANK, [119](#)
 - BANKREF, [119](#)
 - BANKREF_EXTERN, [119](#)
 - c, [151](#)
 - cancel_pending_interrupts, [127](#)
 - CGB_TYPE, [118](#)
 - COMPAT_PALETTE, [123](#)
 - CURRENT_BANK, [119](#)
 - d, [152](#)
 - delay, [128](#)
 - DEVICE_SUPPORTS_COLOR, [118](#)
 - disable_interrupts, [130](#)
 - DISABLE_OAM_DMA, [124](#)
 - DISABLE_RAM, [121](#)
 - DISABLE_RAM_MBC1, [121](#)
 - DISABLE_RAM_MBC5, [122](#)
 - DISABLE_VBL_TRANSFER, [124](#)
 - DISPLAY_OFF, [122](#)
 - display_off, [130](#)
 - DISPLAY_ON, [122](#)
 - DMG_BLACK, [116](#)
 - DMG_DARK_GRAY, [116](#)
 - DMG_LITE_GRAY, [116](#)
 - DMG_PALETTE, [117](#)
 - DMG_TYPE, [118](#)
 - DMG_WHITE, [116](#)
 - e, [153](#)
 - EMPTY_IFLAG, [115](#)
 - enable_interrupts, [129](#)
 - ENABLE_OAM_DMA, [124](#)
 - ENABLE_RAM, [121](#)
 - ENABLE_RAM_MBC1, [121](#)
 - ENABLE_RAM_MBC5, [122](#)
 - ENABLE_VBL_TRANSFER, [124](#)
 - fill_bkg_rect, [151](#)
 - fill_rect, [124](#)
 - fill_win_rect, [151](#)
 - GAMEBOY, [114](#)
 - GBA_DETECTED, [118](#)
 - GBA_NOT_DETECTED, [118](#)
 - get_bkg_data, [133](#)
 - get_bkg_tile_xy, [137](#)

get_bkg_tiles, 136
get_bkg_xy_addr, 131
get_data, 147
get_mode, 128
get_sprite_data, 144
get_sprite_prop, 146
get_sprite_tile, 145
get_tiles, 149
get_vram_byte, 131
get_win_data, 139
get_win_tile_xy, 143
get_win_tiles, 142
get_win_xy_addr, 138
h, 152
HIDE_BKG, 123
HIDE_LEFT_COLUMN, 122
hide_sprite, 147
HIDE_SPRITES, 123
HIDE_WIN, 123
hiramcpy, 131
init_bkg, 150
init_win, 150
int_handler, 124
IO_ERROR, 119
IO_IDLE, 118
IO_RECEIVING, 118
IO_SENDING, 118
J_A, 114
J_B, 114
J_DOWN, 114
J_LEFT, 114
J_RIGHT, 114
J_SELECT, 114
J_START, 114
J_UP, 114
JOY_IFLAG, 116
joypad, 128
joypad_ex, 129
joypad_init, 129
l, 152
LCD_IFLAG, 116
M_DRAWING, 114
M_NO_INTERP, 115
M_NO_SCROLL, 115
M_TEXT_INOUT, 114
M_TEXT_OUT, 114
MAX_HARDWARE_SPRITES, 124
MAXWNDPOSX, 117
MAXWNDPOSY, 117
MGB_TYPE, 118
MINWNDPOSX, 117
MINWNDPOSY, 117
mode, 127
move_bkg, 137
move_sprite, 146
move_win, 143
NINTENDO, 114
nowait_int_handler, 127
OAM_item_t, 124
receive_byte, 128
refresh_OAM, 131
remove_JOY, 125
remove_LCD, 125
remove_SIO, 125
remove_TIM, 125
remove_VBL, 124
reset, 130
S_FLIPX, 115
S_FLIPY, 115
S_PALETTE, 115
S_PRIORITY, 115
SCREENHEIGHT, 117
SCREENWIDTH, 117
scroll_bkg, 138
scroll_sprite, 146
scroll_win, 143
send_byte, 128
set_1bpp_colors, 132
set_1bpp_colors_ex, 131
set_2bpp_palette, 131
set_bkg_1bpp_data, 133
set_bkg_2bpp_data, 123
set_bkg_based_submap, 136
set_bkg_based_tiles, 134
set_bkg_data, 132
set_bkg_submap, 135
set_bkg_tile_xy, 137
set_bkg_tiles, 133
set_data, 147
set_interrupts, 130
set_native_tile_data, 150
SET_SHADOW_OAM_ADDRESS, 145
set_sprite_1bpp_data, 144
set_sprite_2bpp_data, 123
set_sprite_data, 143
set_sprite_prop, 145
set_sprite_tile, 145
set_tile_data, 149
set_tile_map, 123
set_tile_submap, 123
set_tile_xy, 123
set_tiles, 148
set_vram_byte, 131
set_win_1bpp_data, 139
set_win_based_submap, 141
set_win_based_tiles, 140
set_win_data, 138
set_win_submap, 140
set_win_tile_xy, 142
set_win_tiles, 139
shadow_OAM, 153
SHOW_BKG, 123
SHOW_LEFT_COLUMN, 122
SHOW_SPRITES, 123
SHOW_WIN, 123
SIO_IFLAG, 116

- SPRITES_8x16, [123](#)
- SPRITES_8x8, [123](#)
- SWITCH_16_8_MODE_MBC1, [121](#)
- SWITCH_4_32_MODE_MBC1, [121](#)
- SWITCH_RAM, [121](#)
- SWITCH_RAM_MBC1, [120](#)
- SWITCH_RAM_MBC5, [122](#)
- SWITCH_ROM, [120](#)
- SWITCH_ROM_MBC1, [120](#)
- SWITCH_ROM_MBC5, [121](#)
- SWITCH_ROM_MBC5_8M, [121](#)
- SWITCH_ROM_MEGADUCK, [120](#)
- sys_time, [152](#)
- TIM_IFLAG, [116](#)
- VBL_IFLAG, [115](#)
- vmemcpy, [148](#)
- vmemset, [151](#)
- wait_int_handler, [127](#)
- wait_vbl_done, [130](#)
- waitpad, [128](#)
- waitpadup, [129](#)
- gb_decompress
 - gbdecompress.h, [153](#), [156](#)
- gb_decompress_bkg_data
 - gbdecompress.h, [154](#)
- gb_decompress_sprite_data
 - gbdecompress.h, [154](#)
- gb_decompress_win_data
 - gbdecompress.h, [154](#)
- GBA_DETECTED
 - gb.h, [118](#)
- GBA_NOT_DETECTED
 - gb.h, [118](#)
- gbdecompress.h
 - c, [156](#), [157](#)
 - gb_decompress, [153](#), [156](#)
 - gb_decompress_bkg_data, [154](#)
 - gb_decompress_sprite_data, [154](#)
 - gb_decompress_win_data, [154](#)
- gbdk-lib/include/asm/mos6502/provides.h, [72](#)
- gbdk-lib/include/asm/mos6502/stdarg.h, [73](#)
- gbdk-lib/include/asm/mos6502/string.h, [75](#)
- gbdk-lib/include/asm/mos6502/types.h, [86](#)
- gbdk-lib/include/asm/sm83/provides.h, [72](#)
- gbdk-lib/include/asm/sm83/stdarg.h, [74](#)
- gbdk-lib/include/asm/sm83/string.h, [79](#)
- gbdk-lib/include/asm/sm83/types.h, [87](#)
- gbdk-lib/include/asm/types.h, [88](#)
- gbdk-lib/include/asm/z80/provides.h, [73](#)
- gbdk-lib/include/asm/z80/stdarg.h, [74](#)
- gbdk-lib/include/asm/z80/string.h, [82](#)
- gbdk-lib/include/asm/z80/types.h, [90](#)
- gbdk-lib/include/assert.h, [92](#)
- gbdk-lib/include/ctype.h, [92](#)
- gbdk-lib/include/gb/bcd.h, [94](#)
- gbdk-lib/include/gb/bgb_emu.h, [96](#)
- gbdk-lib/include/gb/cgb.h, [96](#)
- gbdk-lib/include/gb/crash_handler.h, [102](#)
- gbdk-lib/include/gb/drawing.h, [102](#)
- gbdk-lib/include/gb/emu_debug.h, [107](#)
- gbdk-lib/include/gb/gb.h, [110](#)
- gbdk-lib/include/gb/gbdecompress.h, [153](#)
- gbdk-lib/include/gb/hardware.h, [157](#)
- gbdk-lib/include/gb/isr.h, [202](#)
- gbdk-lib/include/gb/metasprites.h, [203](#)
- gbdk-lib/include/gb/sgb.h, [220](#)
- gbdk-lib/include/gbdk/bcd.h, [96](#)
- gbdk-lib/include/gbdk/console.h, [223](#)
- gbdk-lib/include/gbdk/emu_debug.h, [107](#)
- gbdk-lib/include/gbdk/far_ptr.h, [224](#)
- gbdk-lib/include/gbdk/font.h, [227](#)
- gbdk-lib/include/gbdk/gbdecompress.h, [156](#)
- gbdk-lib/include/gbdk/gbdk-lib.h, [229](#)
- gbdk-lib/include/gbdk/incbin.h, [229](#)
- gbdk-lib/include/gbdk/metasprites.h, [209](#)
- gbdk-lib/include/gbdk/platform.h, [230](#)
- gbdk-lib/include/gbdk/rledecompress.h, [230](#)
- gbdk-lib/include/gbdk/version.h, [231](#)
- gbdk-lib/include/limits.h, [232](#)
- gbdk-lib/include/msx/hardware.h, [179](#)
- gbdk-lib/include/msx/metasprites.h, [209](#)
- gbdk-lib/include/msx/msx.h, [233](#)
- gbdk-lib/include/nes/hardware.h, [189](#)
- gbdk-lib/include/nes/metasprites.h, [211](#)
- gbdk-lib/include/nes/nes.h, [256](#)
- gbdk-lib/include/nes/rgb_to_nes_macro.h, [284](#)
- gbdk-lib/include/rand.h, [284](#)
- gbdk-lib/include/setjmp.h, [286](#)
- gbdk-lib/include/sms/gbdecompress.h, [156](#)
- gbdk-lib/include/sms/hardware.h, [192](#)
- gbdk-lib/include/sms/metasprites.h, [217](#)
- gbdk-lib/include/sms/sms.h, [287](#)
- gbdk-lib/include/stdarg.h, [75](#)
- gbdk-lib/include/stdatomic.h, [310](#)
- gbdk-lib/include/stdbool.h, [311](#)
- gbdk-lib/include/stddef.h, [311](#)
- gbdk-lib/include/stdint.h, [312](#)
- gbdk-lib/include/stdio.h, [318](#)
- gbdk-lib/include/stdlib.h, [319](#)
- gbdk-lib/include/stdnoreturn.h, [323](#)
- gbdk-lib/include/string.h, [86](#)
- gbdk-lib/include/time.h, [323](#)
- gbdk-lib/include/typeof.h, [324](#)
- gbdk-lib/include/types.h, [91](#)
- get_bkg_data
 - gb.h, [133](#)
- get_bkg_tile_xy
 - gb.h, [137](#)
 - nes.h, [277](#)
- get_bkg_tiles
 - gb.h, [136](#)
 - nes.h, [276](#)
- get_bkg_xy_addr
 - gb.h, [131](#)
 - msx.h, [254](#)
 - nes.h, [270](#)

- sms.h, [309](#)
- get_data
 - gb.h, [147](#)
- get_mode
 - gb.h, [128](#)
 - msx.h, [244](#)
 - nes.h, [267](#)
 - sms.h, [298](#)
- get_sprite_data
 - gb.h, [144](#)
- get_sprite_prop
 - gb.h, [146](#)
 - msx.h, [252](#)
 - nes.h, [280](#)
 - sms.h, [307](#)
- get_sprite_tile
 - gb.h, [145](#)
 - msx.h, [252](#)
 - nes.h, [279](#)
 - sms.h, [307](#)
- get_tiles
 - gb.h, [149](#)
- get_vram_byte
 - gb.h, [131](#)
- get_win_data
 - gb.h, [139](#)
- get_win_tile_xy
 - gb.h, [143](#)
- get_win_tiles
 - gb.h, [142](#)
- get_win_xy_addr
 - gb.h, [138](#)
 - msx.h, [243](#)
 - sms.h, [297](#)
- getchar
 - stdio.h, [319](#)
- getpix
 - drawing.h, [106](#)
- gets
 - stdio.h, [319](#)
- gotogxy
 - drawing.h, [106](#)
- gotoxy
 - console.h, [223](#)
- gprint
 - drawing.h, [104](#)
- gprintf
 - drawing.h, [105](#)
- gprintln
 - drawing.h, [104](#)
- gprintn
 - drawing.h, [105](#)
- GRAPHICS_HEIGHT
 - drawing.h, [103](#)
- GRAPHICS_WIDTH
 - drawing.h, [103](#)
- h
 - _fixed, [65](#)

- gb.h, [152](#)
- msx.h, [254](#)
- sms.h, [309](#)
- hardware.h
 - _AUD3WAVERAM, [176](#)
 - _BIOS, [188](#), [201](#)
 - _HRAM, [176](#)
 - _IO, [176](#)
 - _OAMRAM, [175](#)
 - _RAM, [175](#)
 - _RAMBANK, [175](#)
 - _SCRN0, [175](#)
 - _SCRN1, [175](#)
 - _SRAM, [175](#)
 - _SYSTEM, [189](#), [201](#)
 - _VRAM, [175](#)
 - _VRAM8000, [175](#)
 - _VRAM8800, [175](#)
 - _VRAM9000, [175](#)
 - __BYTES, [162](#), [182](#), [190](#), [195](#)
 - __BYTE_REG, [162](#), [182](#), [190](#), [195](#)
 - __REG, [162](#)
 - AUD1SWEEP_DOWN, [164](#)
 - AUD1SWEEP_LENGTH, [165](#)
 - AUD1SWEEP_TIME, [164](#)
 - AUD1SWEEP_UP, [164](#)
 - AUD3WAVE, [178](#)
 - AUD4POLY_WIDTH_15BIT, [166](#)
 - AUD4POLY_WIDTH_7BIT, [166](#)
 - AUDENA_OFF, [167](#)
 - AUDENA_ON, [167](#)
 - AUDENV_DOWN, [173](#)
 - AUDENV_LENGTH, [173](#)
 - AUDENV_UP, [173](#)
 - AUDENV_VOL, [173](#)
 - AUDHIGH_LENGTH_OFF, [173](#)
 - AUDHIGH_LENGTH_ON, [173](#)
 - AUDHIGH_RESTART, [173](#)
 - AUDLEN_DUTY_12_5, [173](#)
 - AUDLEN_DUTY_25, [173](#)
 - AUDLEN_DUTY_50, [173](#)
 - AUDLEN_DUTY_75, [173](#)
 - AUDLEN_LENGTH, [173](#)
 - AUDTERM_1_LEFT, [166](#)
 - AUDTERM_1_RIGHT, [166](#)
 - AUDTERM_2_LEFT, [166](#)
 - AUDTERM_2_RIGHT, [166](#)
 - AUDTERM_3_LEFT, [166](#)
 - AUDTERM_3_RIGHT, [166](#)
 - AUDTERM_4_LEFT, [166](#)
 - AUDTERM_4_RIGHT, [166](#)
 - AUDVOL_VIN_LEFT, [166](#)
 - AUDVOL_VIN_RIGHT, [166](#)
 - AUDVOL_VOL_LEFT, [166](#)
 - AUDVOL_VOL_RIGHT, [166](#)
 - BCPD_REG, [179](#)
 - BCPS_REG, [179](#)
 - BCPSF_AUTOINC, [172](#)

BGP_REG, 178
bkg_scroll_x, 192
bkg_scroll_y, 192
BKGF_BANK0, 170
BKGF_BANK1, 170
BKGF_CGB_PAL0, 170
BKGF_CGB_PAL1, 171
BKGF_CGB_PAL2, 171
BKGF_CGB_PAL3, 171
BKGF_CGB_PAL4, 171
BKGF_CGB_PAL5, 171
BKGF_CGB_PAL6, 171
BKGF_CGB_PAL7, 171
BKGF_PRI, 170
BKGF_XFLIP, 170
BKGF_YFLIP, 170
DEVICE_SCREEN_BUFFER_HEIGHT, 174, 191
DEVICE_SCREEN_BUFFER_WIDTH, 174, 191
DEVICE_SCREEN_HEIGHT, 174, 191
DEVICE_SCREEN_MAP_ENTRY_SIZE, 174, 191
DEVICE_SCREEN_PX_HEIGHT, 175, 188, 192, 201
DEVICE_SCREEN_PX_WIDTH, 175, 188, 192, 200
DEVICE_SCREEN_WIDTH, 174, 191
DEVICE_SCREEN_X_OFFSET, 174, 191
DEVICE_SCREEN_Y_OFFSET, 174, 191
DEVICE_SPRITE_PX_OFFSET_X, 175, 192
DEVICE_SPRITE_PX_OFFSET_Y, 175, 192
DEVICE_WINDOW_PX_OFFSET_X, 175, 192
DEVICE_WINDOW_PX_OFFSET_Y, 175, 192
DIV_REG, 176
DMA_REG, 178
HDMA1_REG, 178
HDMA2_REG, 179
HDMA3_REG, 179
HDMA4_REG, 179
HDMA5_REG, 179
HDMA5F_BUSY, 171
HDMA5F_MODE_GP, 171
HDMA5F_MODE_HBL, 171
IE_REG, 179
IEF_HILO, 172
IEF_SERIAL, 172
IEF_STAT, 172
IEF_TIMER, 172
IEF_VBLANK, 172
IF_REG, 176
JOY_P1_DOWN, 186, 199
JOY_P1_LATCH, 183, 195
JOY_P1_LEFT, 186, 199
JOY_P1_LIGHT, 187, 200
JOY_P1_RIGHT, 186, 199
JOY_P1_SW1, 187, 199
JOY_P1_SW2, 187, 199
JOY_P1_TRIGGER, 187, 199
JOY_P1_UP, 186, 199
JOY_P2_DOWN, 187, 199
JOY_P2_LATCH, 183, 196
JOY_P2_LEFT, 187, 199
JOY_P2_LIGHT, 187, 200
JOY_P2_RIGHT, 187, 200
JOY_P2_SW1, 187, 200
JOY_P2_SW2, 187, 200
JOY_P2_TRIGGER, 187, 200
JOY_P2_UP, 187, 199
JOY_RESET, 187, 200
KEY1_REG, 178
KEY1F_DBLSPEED, 170
KEY1F_PREPARE, 170
LCDC_REG, 178
LCDCF_B_BG8000, 168
LCDCF_B_BG9C00, 168
LCDCF_B_BGON, 168
LCDCF_B_OBJ16, 168
LCDCF_B_OBJON, 168
LCDCF_B_ON, 168
LCDCF_B_WIN9C00, 168
LCDCF_B_WINON, 168
LCDCF_BG8000, 167
LCDCF_BG8800, 167
LCDCF_BG9800, 167
LCDCF_BG9C00, 167
LCDCF_BGOFF, 168
LCDCF_BGON, 168
LCDCF_OBJ16, 167
LCDCF_OBJ8, 167
LCDCF_OBJOFF, 167
LCDCF_OBJON, 167
LCDCF_OFF, 167
LCDCF_ON, 167
LCDCF_WIN9800, 167
LCDCF_WIN9C00, 167
LCDCF_WINOFF, 167
LCDCF_WINON, 167
LY_REG, 178
LYC_REG, 178
MEMCTL_BASEOFF, 182, 195
MEMCTL_BASEON, 182, 195
MEMCTL_CROMOFF, 182, 195
MEMCTL_CROMON, 182, 195
MEMCTL_EXTOFF, 183, 195
MEMCTL_EXTON, 182, 195
MEMCTL_JOYOFF, 182, 195
MEMCTL_JOYON, 182, 195
MEMCTL_RAMOFF, 182, 195
MEMCTL_RAMON, 182, 195
MEMCTL_ROMOFF, 182, 195
MEMCTL_ROMON, 182, 195
NR10_REG, 176
NR11_REG, 176
NR12_REG, 176
NR13_REG, 177
NR14_REG, 177
NR21_REG, 177
NR22_REG, 177

NR23_REG, [177](#)
NR24_REG, [177](#)
NR30_REG, [177](#)
NR31_REG, [177](#)
NR32_REG, [177](#)
NR33_REG, [177](#)
NR34_REG, [177](#)
NR41_REG, [177](#)
NR42_REG, [177](#)
NR43_REG, [177](#)
NR44_REG, [177](#)
NR50_REG, [177](#)
NR51_REG, [177](#)
NR52_REG, [177](#)
OAMADDR, [191](#)
OAMDATA, [191](#)
OAMDMA, [191](#)
OAMF_BANK0, [174](#)
OAMF_BANK1, [174](#)
OAMF_CGB_PAL0, [174](#)
OAMF_CGB_PAL1, [174](#)
OAMF_CGB_PAL2, [174](#)
OAMF_CGB_PAL3, [174](#)
OAMF_CGB_PAL4, [174](#)
OAMF_CGB_PAL5, [174](#)
OAMF_CGB_PAL6, [174](#)
OAMF_CGB_PAL7, [174](#)
OAMF_PAL0, [173](#)
OAMF_PAL1, [173](#)
OAMF_PALMASK, [174](#)
OAMF_PRI, [173](#)
OAMF_XFLIP, [173](#)
OAMF_YFLIP, [173](#)
OBP0_REG, [178](#)
OBP1_REG, [178](#)
OCPD_REG, [179](#)
OCPS_REG, [179](#)
OCPSF_AUTOINC, [172](#)
P1_REG, [176](#)
P1F_0, [163](#)
P1F_1, [163](#)
P1F_2, [163](#)
P1F_3, [163](#)
P1F_4, [163](#)
P1F_5, [163](#)
P1F_GET_BTN, [163](#)
P1F_GET_DPAD, [163](#)
P1F_GET_NONE, [163](#)
PCM12_REG, [179](#)
PCM34_REG, [179](#)
PCM_SAMPLE, [178](#)
PPUADDR, [191](#)
PPUCTRL, [190](#)
PPUCTRL_BG_CHR, [190](#)
PPUCTRL_INC32, [190](#)
PPUCTRL_NMI, [190](#)
PPUCTRL_SPR_8X16, [190](#)
PPUCTRL_SPR_8X8, [190](#)
PPUCTRL_SPR_CHR, [190](#)
PPUDATA, [191](#)
PPUMASK, [190](#)
PPUMASK_BLUE, [190](#)
PPUMASK_GREEN, [190](#)
PPUMASK_MONOCHROME, [191](#)
PPUMASK_RED, [190](#)
PPUMASK_SHOW_BG, [191](#)
PPUMASK_SHOW_BG_LC, [191](#)
PPUMASK_SHOW_SPR, [190](#)
PPUMASK_SHOW_SPR_LC, [191](#)
PPUSCROLL, [191](#)
PPUSTATUS, [191](#)
PSG_CH0, [183](#), [196](#)
PSG_CH1, [183](#), [196](#)
PSG_CH2, [183](#), [196](#)
PSG_CH3, [183](#), [196](#)
PSG_LATCH, [183](#), [196](#)
PSG_VOLUME, [183](#), [196](#)
R0_CB_INPUT, [183](#)
R0_CB_OUTPUT, [183](#)
R0_DEFAULT, [183](#), [197](#)
R0_ES, [184](#), [197](#)
R0_ES_OFF, [184](#), [197](#)
R0_HSCRL, [196](#)
R0_HSCRL_INH, [196](#)
R0_IE1, [184](#), [197](#)
R0_IE1_OFF, [184](#), [197](#)
R0_IE2, [184](#)
R0_IE2_OFF, [183](#)
R0_LCB, [196](#)
R0_NO_LCB, [196](#)
R0_SCR_MODE1, [184](#)
R0_SCR_MODE2, [184](#)
R0_SCR_MODE3, [184](#)
R0_SS, [197](#)
R0_SS_OFF, [197](#)
R0_VSCRL, [196](#)
R0_VSCRL_INH, [196](#)
R10_INT_EVERY, [186](#), [199](#)
R10_INT_OFF, [186](#), [199](#)
R1_DEFAULT, [184](#), [197](#)
R1_DISP_OFF, [184](#), [197](#)
R1_DISP_ON, [184](#), [197](#)
R1_IE, [184](#), [197](#)
R1_IE_OFF, [184](#), [197](#)
R1_SCR_MODE1, [184](#)
R1_SCR_MODE2, [184](#)
R1_SCR_MODE3, [184](#)
R1_SPR_16X16, [185](#)
R1_SPR_8X16, [197](#)
R1_SPR_8X8, [184](#), [197](#)
R1_SPR_MAG, [185](#)
R1_SPR_MAG_OFF, [185](#)
R2_MAP_0x0000, [185](#), [198](#)
R2_MAP_0x0800, [185](#), [198](#)
R2_MAP_0x1000, [185](#), [198](#)
R2_MAP_0x1800, [185](#), [198](#)

R2_MAP_0x2000, [185](#), [198](#)
R2_MAP_0x2800, [185](#), [198](#)
R2_MAP_0x3000, [185](#), [197](#)
R2_MAP_0x3800, [185](#), [197](#)
R5_SAT_0x3F00, [185](#), [198](#)
R5_SAT_MASK, [185](#), [198](#)
R6_BANK0, [186](#), [198](#)
R6_BANK1, [186](#), [198](#)
R6_DATA_0x0000, [186](#), [198](#)
R6_DATA_0x2000, [186](#), [198](#)
R7_COLOR_MASK, [186](#), [199](#)
RAMCTL_BANK, [200](#)
RAMCTL_PROT, [200](#)
RAMCTL_RAM, [200](#)
RAMCTL_RO, [200](#)
RAMCTL_ROM, [200](#)
rAUD1ENV, [165](#)
rAUD1HIGH, [165](#)
rAUD1LEN, [165](#)
rAUD1LOW, [165](#)
rAUD1SWEEP, [164](#)
rAUD2ENV, [165](#)
rAUD2HIGH, [165](#)
rAUD2LEN, [165](#)
rAUD2LOW, [165](#)
rAUD3ENA, [165](#)
rAUD3HIGH, [165](#)
rAUD3LEN, [165](#)
rAUD3LEVEL, [165](#)
rAUD3LOW, [165](#)
rAUD4ENV, [165](#)
rAUD4GO, [166](#)
rAUD4LEN, [165](#)
rAUD4POLY, [165](#)
rAUDENA, [167](#)
rAUDTERM, [166](#)
rAUDVOL, [166](#)
rBCPD, [172](#)
rBCPS, [172](#)
rBGP, [169](#)
rDIV, [163](#)
rDMA, [169](#)
rHDMA1, [171](#)
rHDMA2, [171](#)
rHDMA3, [171](#)
rHDMA4, [171](#)
rHDMA5, [171](#)
rIE, [172](#)
rIF, [164](#)
rKEY1, [170](#)
rLCDC, [167](#)
rLY, [169](#)
rLYC, [169](#)
rOBP0, [169](#)
rOBP1, [170](#)
rOCPD, [172](#)
rOCPS, [172](#)
rP1, [163](#)
RP_REG, [179](#)
rPCM12, [172](#)
rPCM34, [172](#)
RPF_DATAIN, [171](#)
RPF_ENREAD, [171](#)
RPF_WRITE_HI, [172](#)
RPF_WRITE_LO, [172](#)
rRAMB, [176](#)
rRAMG, [176](#)
rROMB0, [176](#)
rROMB1, [176](#)
rRP, [171](#)
rSB, [163](#)
rSC, [163](#)
rSCX, [169](#)
rSCY, [169](#)
rSMBK, [172](#)
rSPD, [170](#)
rSTAT, [168](#)
rSVBK, [172](#)
rTAC, [163](#)
rTIMA, [163](#)
rTMA, [163](#)
rVBK, [170](#)
rWX, [170](#)
rWY, [170](#)
SB_REG, [176](#)
SC_REG, [176](#)
SCX_REG, [178](#)
SCY_REG, [178](#)
shadow_PPUCTRL, [192](#)
shadow_PPUMASK, [192](#)
shadow_VDP_R0, [188](#), [201](#)
shadow_VDP_R1, [188](#), [201](#)
shadow_VDP_R10, [188](#), [201](#)
shadow_VDP_R2, [188](#), [201](#)
shadow_VDP_R3, [188](#), [201](#)
shadow_VDP_R4, [188](#), [201](#)
shadow_VDP_R5, [188](#), [201](#)
shadow_VDP_R6, [188](#), [201](#)
shadow_VDP_R7, [188](#), [201](#)
shadow_VDP_R8, [188](#), [201](#)
shadow_VDP_R9, [188](#), [201](#)
shadow_VDP_RBORDER, [188](#), [201](#)
shadow_VDP_RSCX, [188](#), [201](#)
shadow_VDP_RSCY, [188](#), [201](#)
SIOF_B_CLOCK, [164](#)
SIOF_B_SPEED, [164](#)
SIOF_B_XFER_START, [164](#)
SIOF_CLOCK_EXT, [164](#)
SIOF_CLOCK_INT, [164](#)
SIOF_SPEED_1X, [164](#)
SIOF_SPEED_32X, [164](#)
SIOF_XFER_START, [164](#)
STAT_REG, [178](#)
STATF_9_SPR, [183](#), [196](#)
STATF_B_BUSY, [169](#)
STATF_B_LYC, [169](#)

- STATF_B_LYCF, [169](#)
- STATF_B_MODE00, [169](#)
- STATF_B_MODE01, [169](#)
- STATF_B_MODE10, [169](#)
- STATF_B_OAM, [169](#)
- STATF_B_VBL, [169](#)
- STATF_BUSY, [169](#)
- STATF_HBL, [168](#)
- STATF_INT_VBL, [183](#), [196](#)
- STATF_LCD, [169](#)
- STATF_LYC, [168](#)
- STATF_LYCF, [168](#)
- STATF_MODE00, [168](#)
- STATF_MODE01, [168](#)
- STATF_MODE10, [168](#)
- STATF_OAM, [169](#)
- STATF_SPR_COLL, [183](#), [196](#)
- STATF_VBL, [168](#)
- SVBK_REG, [179](#)
- SYSTEM_NTSC, [187](#), [200](#)
- SYSTEM_PAL, [187](#), [200](#)
- TAC_REG, [176](#)
- TACF_16KHZ, [164](#)
- TACF_262KHZ, [164](#)
- TACF_4KHZ, [164](#)
- TACF_65KHZ, [164](#)
- TACF_START, [163](#)
- TACF_STOP, [163](#)
- TIMA_REG, [176](#)
- TMA_REG, [176](#)
- VBK_ATTRIBUTES, [170](#), [187](#), [200](#)
- VBK_BANK_0, [170](#)
- VBK_BANK_1, [170](#)
- VBK_REG, [178](#)
- VBK_TILES, [170](#), [187](#), [200](#)
- VDP_ATTR_SHIFT, [189](#), [202](#)
- VDP_R0, [183](#), [196](#)
- VDP_R1, [184](#), [197](#)
- VDP_R10, [186](#), [199](#)
- VDP_R2, [185](#), [197](#)
- VDP_R3, [185](#), [198](#)
- VDP_R4, [185](#), [198](#)
- VDP_R5, [185](#), [198](#)
- VDP_R6, [185](#), [198](#)
- VDP_R7, [186](#), [198](#)
- VDP_R8, [186](#), [199](#)
- VDP_R9, [186](#), [199](#)
- VDP_RBORDER, [186](#), [198](#)
- VDP_REG_MASK, [183](#), [196](#)
- VDP_RSCX, [186](#), [199](#)
- VDP_RSCY, [186](#), [199](#)
- VDP_SAT_TERM, [187](#), [200](#)
- WX_REG, [178](#)
- WY_REG, [178](#)
- HDMA1_REG
 - hardware.h, [178](#)
- HDMA2_REG
 - hardware.h, [179](#)
- HDMA3_REG
 - hardware.h, [179](#)
- HDMA4_REG
 - hardware.h, [179](#)
- HDMA5_REG
 - hardware.h, [179](#)
- HDMA5F_BUSY
 - hardware.h, [171](#)
- HDMA5F_MODE_GP
 - hardware.h, [171](#)
- HDMA5F_MODE_HBL
 - hardware.h, [171](#)
- HIDE_BKG
 - gb.h, [123](#)
 - msx.h, [240](#)
 - nes.h, [265](#)
 - sms.h, [294](#)
- HIDE_LEFT_COLUMN
 - gb.h, [122](#)
 - msx.h, [240](#)
 - nes.h, [265](#)
 - sms.h, [294](#)
- hide metasprite
 - metasprites.h, [208](#), [211](#), [216](#), [219](#)
- hide_sprite
 - gb.h, [147](#)
 - msx.h, [253](#)
 - nes.h, [281](#)
 - sms.h, [308](#)
- HIDE_SPRITES
 - gb.h, [123](#)
 - msx.h, [240](#)
 - nes.h, [265](#)
 - sms.h, [294](#)
- hide_sprites_range
 - metasprites.h, [206](#), [210](#), [213](#), [218](#)
- HIDE_WIN
 - gb.h, [123](#)
 - msx.h, [240](#)
 - sms.h, [294](#)
- hramcpy
 - gb.h, [131](#)
- IE_REG
 - hardware.h, [179](#)
- IEF_HILO
 - hardware.h, [172](#)
- IEF_SERIAL
 - hardware.h, [172](#)
- IEF_STAT
 - hardware.h, [172](#)
- IEF_TIMER
 - hardware.h, [172](#)
- IEF_VBLANK
 - hardware.h, [172](#)
- IF_REG
 - hardware.h, [176](#)
- INCBIN
 - incbin.h, [230](#)

incbin.h
 BANK, 229
 INCBIN, 230
 INCBIN_EXTERN, 229
 INCBIN_SIZE, 229
INCBIN_EXTERN
 incbin.h, 229
INCBIN_SIZE
 incbin.h, 229
init_bkg
 gb.h, 150
 nes.h, 283
init_win
 gb.h, 150
initarand
 rand.h, 286
initrand
 rand.h, 285
INT16
 types.h, 87, 88, 91
INT16_C
 stdint.h, 315
INT16_MAX
 stdint.h, 313
INT16_MIN
 stdint.h, 313
int16_t
 stdint.h, 316
INT32
 types.h, 87, 88, 91
INT32_C
 stdint.h, 316
INT32_MAX
 stdint.h, 314
INT32_MIN
 stdint.h, 313
int32_t
 stdint.h, 316
INT8
 types.h, 87, 88, 91
INT8_C
 stdint.h, 315
INT8_MAX
 stdint.h, 313
INT8_MIN
 stdint.h, 313
int8_t
 stdint.h, 316
INT_FAST16_MAX
 stdint.h, 314
INT_FAST16_MIN
 stdint.h, 314
int_fast16_t
 stdint.h, 317
INT_FAST32_MAX
 stdint.h, 315
INT_FAST32_MIN
 stdint.h, 314
int_fast32_t
 stdint.h, 317
INT_FAST8_MAX
 stdint.h, 314
INT_FAST8_MIN
 stdint.h, 314
int_fast8_t
 stdint.h, 317
int_handler
 gb.h, 124
 msx.h, 243
 sms.h, 297
INT_LEAST16_MAX
 stdint.h, 314
INT_LEAST16_MIN
 stdint.h, 314
int_least16_t
 stdint.h, 317
INT_LEAST32_MAX
 stdint.h, 314
INT_LEAST32_MIN
 stdint.h, 314
int_least32_t
 stdint.h, 317
INT_LEAST8_MAX
 stdint.h, 314
INT_LEAST8_MIN
 stdint.h, 314
int_least8_t
 stdint.h, 317
INT_MAX
 limits.h, 232
INT_MIN
 limits.h, 232
INTERRUPT
 types.h, 89
INTMAX_C
 stdint.h, 316
INTMAX_MAX
 stdint.h, 315
INTMAX_MIN
 stdint.h, 315
intmax_t
 stdint.h, 317
INTPTR_MAX
 stdint.h, 315
INTPTR_MIN
 stdint.h, 315
intptr_t
 stdint.h, 317
IO_ERROR
 gb.h, 119
IO_IDLE
 gb.h, 118
IO_RECEIVING
 gb.h, 118
IO_SENDING
 gb.h, 118

- isalpha
 - ctype.h, 93
- isdigit
 - ctype.h, 93
- islower
 - ctype.h, 93
- isr.h
 - ISR_NESTED_VECTOR, 203
 - isr_nested_vector_t, 203
 - ISR_VECTOR, 202
 - isr_vector_t, 203
 - VECTOR_JOYPAD, 202
 - VECTOR_SERIAL, 202
 - VECTOR_STAT, 202
 - VECTOR_TIMER, 202
- ISR_NESTED_VECTOR
 - isr.h, 203
- isr_nested_vector_t, 66
 - func, 66
 - isr.h, 203
 - opcode, 66
- ISR_VECTOR
 - isr.h, 202
- isr_vector_t, 66
 - func, 67
 - isr.h, 203
 - opcode, 67
- isspace
 - ctype.h, 93
- isupper
 - ctype.h, 93
- itoa
 - stdlib.h, 321
- iyh
 - msx.h, 254
 - sms.h, 309
- iyi
 - metasprites.h, 211, 220
 - msx.h, 254
 - sms.h, 309
- J_A
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
 - sms.h, 291
- J_B
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
 - sms.h, 291
- J_DOWN
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
 - sms.h, 291
- J_LEFT
 - gb.h, 114
 - msx.h, 237
- nes.h, 260
- sms.h, 291
- J_RIGHT
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
 - sms.h, 291
- J_SELECT
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
- J_START
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
- J_UP
 - gb.h, 114
 - msx.h, 237
 - nes.h, 260
 - sms.h, 291
- jmp_buf
 - setjmp.h, 287
- joy0
 - joypads_t, 68
- joy1
 - joypads_t, 68
- joy2
 - joypads_t, 68
- joy3
 - joypads_t, 68
- JOY_IFLAG
 - gb.h, 116
 - msx.h, 239
 - sms.h, 293
- JOY_P1_DOWN
 - hardware.h, 186, 199
- JOY_P1_LATCH
 - hardware.h, 183, 195
- JOY_P1_LEFT
 - hardware.h, 186, 199
- JOY_P1_LIGHT
 - hardware.h, 187, 200
- JOY_P1_RIGHT
 - hardware.h, 186, 199
- JOY_P1_SW1
 - hardware.h, 187, 199
- JOY_P1_SW2
 - hardware.h, 187, 199
- JOY_P1_TRIGGER
 - hardware.h, 187, 199
- JOY_P1_UP
 - hardware.h, 186, 199
- JOY_P2_DOWN
 - hardware.h, 187, 199
- JOY_P2_LATCH
 - hardware.h, 183, 196
- JOY_P2_LEFT
 - hardware.h, 187, 199

JOY_P2_LIGHT
 hardware.h, [187](#), [200](#)

JOY_P2_RIGHT
 hardware.h, [187](#), [200](#)

JOY_P2_SW1
 hardware.h, [187](#), [200](#)

JOY_P2_SW2
 hardware.h, [187](#), [200](#)

JOY_P2_TRIGGER
 hardware.h, [187](#), [200](#)

JOY_P2_UP
 hardware.h, [187](#), [199](#)

JOY_RESET
 hardware.h, [187](#), [200](#)

joypad
 gb.h, [128](#)
 msx.h, [246](#)
 nes.h, [267](#)
 sms.h, [300](#)

joypad_ex
 gb.h, [129](#)
 msx.h, [246](#)
 nes.h, [269](#)
 sms.h, [300](#)

joypad_init
 gb.h, [129](#)
 msx.h, [246](#)
 nes.h, [268](#)
 sms.h, [300](#)

joypads
 joypads_t, [68](#)

joypads_t, [67](#)
 joy0, [68](#)
 joy1, [68](#)
 joy2, [68](#)
 joy3, [68](#)
 joypads, [68](#)
 npads, [68](#)

KEY1_REG
 hardware.h, [178](#)

KEY1F_DBLSPED
 hardware.h, [170](#)

KEY1F_PREPARE
 hardware.h, [170](#)

I
 _fixed, [65](#)
 gb.h, [152](#)
 msx.h, [254](#)
 sms.h, [309](#)

labs
 stdlib.h, [320](#)

LCD_IFLAG
 gb.h, [116](#)
 msx.h, [238](#)
 sms.h, [292](#)

LCDC_REG
 hardware.h, [178](#)

LCDCF_B_BG8000
 hardware.h, [168](#)

LCDCF_B_BG9C00
 hardware.h, [168](#)

LCDCF_B_BGON
 hardware.h, [168](#)

LCDCF_B_OBJ16
 hardware.h, [168](#)

LCDCF_B_OBJON
 hardware.h, [168](#)

LCDCF_B_ON
 hardware.h, [168](#)

LCDCF_B_WIN9C00
 hardware.h, [168](#)

LCDCF_B_WINON
 hardware.h, [168](#)

LCDCF_BG8000
 hardware.h, [167](#)

LCDCF_BG8800
 hardware.h, [167](#)

LCDCF_BG9800
 hardware.h, [167](#)

LCDCF_BG9C00
 hardware.h, [167](#)

LCDCF_BGOFF
 hardware.h, [168](#)

LCDCF_BGON
 hardware.h, [168](#)

LCDCF_OBJ16
 hardware.h, [167](#)

LCDCF_OBJ8
 hardware.h, [167](#)

LCDCF_OBJOFF
 hardware.h, [167](#)

LCDCF_OBJON
 hardware.h, [167](#)

LCDCF_OFF
 hardware.h, [167](#)

LCDCF_ON
 hardware.h, [167](#)

LCDCF_WIN9800
 hardware.h, [167](#)

LCDCF_WIN9C00
 hardware.h, [167](#)

LCDCF_WINOFF
 hardware.h, [167](#)

LCDCF_WINON
 hardware.h, [167](#)

limits.h
 CHAR_BIT, [232](#)
 CHAR_MAX, [232](#)
 CHAR_MIN, [232](#)
 INT_MAX, [232](#)
 INT_MIN, [232](#)
 LONG_MAX, [233](#)
 LONG_MIN, [233](#)
 SCHAR_MAX, [232](#)
 SCHAR_MIN, [232](#)

- SHRT_MAX, [232](#)
- SHRT_MIN, [233](#)
- UCHAR_MAX, [232](#)
- UINT_MAX, [233](#)
- UINT_MIN, [233](#)
- ULONG_MAX, [233](#)
- ULONG_MIN, [233](#)
- USHRT_MAX, [233](#)
- USHRT_MIN, [233](#)
- line
 - drawing.h, [106](#)
- List of gbdk fonts, [64](#)
 - font_ibm, [64](#)
 - font_ibm_fixed, [64](#)
 - font_italic, [64](#)
 - font_min, [64](#)
 - font_spect, [64](#)
- LONG_MAX
 - limits.h, [233](#)
- LONG_MIN
 - limits.h, [233](#)
- longjmp
 - setjmp.h, [287](#)
- LTGREY
 - drawing.h, [104](#)
- ltoa
 - stdlib.h, [321](#)
- LWORD
 - types.h, [90](#)
- LY_REG
 - hardware.h, [178](#)
- LYC_REG
 - hardware.h, [178](#)
- M_DRAWING
 - gb.h, [114](#)
 - nes.h, [260](#)
- M_FILL
 - drawing.h, [104](#)
- M_NO_INTERP
 - gb.h, [115](#)
 - msx.h, [238](#)
 - nes.h, [261](#)
 - sms.h, [292](#)
- M_NO_SCROLL
 - gb.h, [115](#)
 - msx.h, [237](#)
 - nes.h, [261](#)
 - sms.h, [292](#)
- M_NOFILL
 - drawing.h, [104](#)
- M_TEXT_INOUT
 - gb.h, [114](#)
 - msx.h, [237](#)
 - nes.h, [261](#)
 - sms.h, [291](#)
- M_TEXT_OUT
 - gb.h, [114](#)
 - msx.h, [237](#)
- nes.h, [261](#)
- sms.h, [291](#)
- MAKE_BCD
 - bcd.h, [95](#)
- malloc
 - stdlib.h, [322](#)
- MAX_HARDWARE_SPRITES
 - gb.h, [124](#)
 - msx.h, [243](#)
 - nes.h, [266](#)
 - sms.h, [297](#)
- MAXWNDPOSX
 - gb.h, [117](#)
 - msx.h, [239](#)
 - sms.h, [293](#)
- MAXWNDPOSY
 - gb.h, [117](#)
 - msx.h, [239](#)
 - sms.h, [293](#)
- memcmp
 - string.h, [78](#), [82](#), [86](#)
- memcpy
 - string.h, [76](#), [80](#), [83](#)
- MEMCTL_BASEOFF
 - hardware.h, [182](#), [195](#)
- MEMCTL_BASEON
 - hardware.h, [182](#), [195](#)
- MEMCTL_CROMOFF
 - hardware.h, [182](#), [195](#)
- MEMCTL_CROMON
 - hardware.h, [182](#), [195](#)
- MEMCTL_EXTOFF
 - hardware.h, [183](#), [195](#)
- MEMCTL_EXTON
 - hardware.h, [182](#), [195](#)
- MEMCTL_JOYOFF
 - hardware.h, [182](#), [195](#)
- MEMCTL_JOYON
 - hardware.h, [182](#), [195](#)
- MEMCTL_RAMOFF
 - hardware.h, [182](#), [195](#)
- MEMCTL_RAMON
 - hardware.h, [182](#), [195](#)
- MEMCTL_ROMOFF
 - hardware.h, [182](#), [195](#)
- MEMCTL_ROMON
 - hardware.h, [182](#), [195](#)
- memmove
 - string.h, [76](#), [80](#), [84](#)
- memset
 - string.h, [76](#), [80](#), [84](#)
- METASPR_ITEM
 - metasprites.h, [205](#), [209](#), [213](#), [218](#)
- METASPR_TERM
 - metasprites.h, [205](#), [210](#), [213](#), [218](#)
- metasprite_end
 - metasprites.h, [205](#), [209](#), [213](#), [218](#)
- metasprite_t, [68](#)

- dtile, [70](#)
- dx, [70](#)
- dy, [70](#)
- metasprites.h, [205](#), [210](#), [213](#), [218](#)
- props, [70](#)
- metasprites.h
 - __current_base_tile, [209](#), [211](#), [216](#), [220](#)
 - __current_metasprite, [208](#), [211](#), [216](#), [219](#)
 - __render_shadow_OAM, [209](#), [211](#), [216](#), [220](#)
- c, [209](#)
- hide_metasprite, [208](#), [211](#), [216](#), [219](#)
- hide_sprites_range, [206](#), [210](#), [213](#), [218](#)
- iyI, [211](#), [220](#)
- METASPR_ITEM, [205](#), [209](#), [213](#), [218](#)
- METASPR_TERM, [205](#), [210](#), [213](#), [218](#)
- metasprite_end, [205](#), [209](#), [213](#), [218](#)
- metasprite_t, [205](#), [210](#), [213](#), [218](#)
- move_metasprite, [206](#), [210](#), [214](#), [219](#)
- move_metasprite_hflip, [207](#), [215](#)
- move_metasprite_hvflip, [208](#), [215](#)
- move_metasprite_vflip, [206](#), [214](#)
- mfont_handle
 - font.h, [228](#)
- MGB_TYPE
 - gb.h, [118](#)
- MINWNDPOSX
 - gb.h, [117](#)
 - msx.h, [239](#)
 - sms.h, [293](#)
- MINWNDPOSY
 - gb.h, [117](#)
 - msx.h, [239](#)
 - sms.h, [293](#)
- mode
 - gb.h, [127](#)
 - msx.h, [244](#)
 - nes.h, [267](#)
 - sms.h, [298](#)
- move_bkg
 - gb.h, [137](#)
 - msx.h, [245](#)
 - nes.h, [277](#)
 - sms.h, [299](#)
- move_metasprite
 - metasprites.h, [206](#), [210](#), [214](#), [219](#)
- move_metasprite_hflip
 - metasprites.h, [207](#), [215](#)
- move_metasprite_hvflip
 - metasprites.h, [208](#), [215](#)
- move_metasprite_vflip
 - metasprites.h, [206](#), [214](#)
- move_sprite
 - gb.h, [146](#)
 - msx.h, [252](#)
 - nes.h, [280](#)
 - sms.h, [307](#)
- move_win
 - gb.h, [143](#)
- MSX
 - msx.h, [237](#)
- msx.h
 - __READ_VDP_REG, [238](#)
 - __WRITE_VDP_REG, [238](#)
 - _current_1bpp_colors, [255](#)
 - _current_2bpp_palette, [255](#)
 - _current_bank, [255](#)
 - _map_tile_offset, [255](#)
 - _shadow_OAM_OFF, [255](#)
 - _shadow_OAM_base, [255](#)
 - _submap_tile_offset, [255](#)
 - add_JOY, [245](#)
 - add_LCD, [245](#)
 - add_SIO, [245](#)
 - add_TIM, [245](#)
 - add_VBL, [245](#)
 - BANK, [241](#)
 - BANKREF, [241](#)
 - BANKREF_EXTERN, [241](#)
 - c, [254](#)
 - cancel_pending_interrupts, [245](#)
 - COMPAT_PALETTE, [242](#)
 - cpu_fast, [247](#)
 - CURRENT_BANK, [240](#)
 - d, [254](#)
 - delay, [246](#)
 - DEVICE_SUPPORTS_COLOR, [240](#)
 - DISABLE_RAM, [242](#)
 - DISABLE_VBL_TRANSFER, [243](#)
 - DISPLAY_OFF, [239](#)
 - display_off, [245](#)
 - DISPLAY_ON, [239](#)
 - e, [254](#)
 - EMPTY_IFLAG, [238](#)
 - ENABLE_RAM, [242](#)
 - ENABLE_VBL_TRANSFER, [243](#)
 - fill_bkg_rect, [243](#)
 - fill_rect, [251](#)
 - fill_win_rect, [243](#)
 - get_bkg_xy_addr, [254](#)
 - get_mode, [244](#)
 - get_sprite_prop, [252](#)
 - get_sprite_tile, [252](#)
 - get_win_xy_addr, [243](#)
 - h, [254](#)
 - HIDE_BKG, [240](#)
 - HIDE_LEFT_COLUMN, [240](#)
 - hide_sprite, [253](#)
 - HIDE_SPRITES, [240](#)
 - HIDE_WIN, [240](#)
 - int_handler, [243](#)
 - iyh, [254](#)
 - iyI, [254](#)
 - J_A, [237](#)
 - J_B, [237](#)
 - J_DOWN, [237](#)
 - J_LEFT, [237](#)

- J_RIGHT, [237](#)
- J_SELECT, [237](#)
- J_START, [237](#)
- J_UP, [237](#)
- JOY_IFLAG, [239](#)
- joypad, [246](#)
- joypad_ex, [246](#)
- joypad_init, [246](#)
- I, [254](#)
- LCD_IFLAG, [238](#)
- M_NO_INTERP, [238](#)
- M_NO_SCROLL, [237](#)
- M_TEXT_INOUT, [237](#)
- M_TEXT_OUT, [237](#)
- MAX_HARDWARE_SPRITES, [243](#)
- MAXWNDPOSX, [239](#)
- MAXWNDPOSY, [239](#)
- MINWNDPOSX, [239](#)
- MINWNDPOSY, [239](#)
- mode, [244](#)
- move_bkg, [245](#)
- move_sprite, [252](#)
- MSX, [237](#)
- OAM_item_t, [243](#)
- refresh_OAM, [246](#)
- remove_JOY, [245](#)
- remove_LCD, [244](#)
- remove_SIO, [245](#)
- remove_TIM, [244](#)
- remove_VBL, [244](#)
- S_FLIPX, [238](#)
- S_FLIPY, [238](#)
- S_PALETTE, [238](#)
- S_PRIORITY, [238](#)
- SCREENHEIGHT, [239](#)
- SCREENWIDTH, [239](#)
- scroll_bkg, [245](#)
- scroll_sprite, [253](#)
- set_1bpp_colors, [248](#)
- set_2bpp_palette, [248](#)
- set_attributed_tile_xy, [253](#)
- set_bkg_1bpp_data, [248](#)
- set_bkg_4bpp_data, [247](#)
- set_bkg_based_submap, [251](#)
- set_bkg_based_tiles, [249](#)
- set_bkg_data, [248](#)
- set_bkg_palette, [242](#)
- set_bkg_palette_entry, [242](#)
- set_bkg_submap, [249](#)
- set_bkg_tile_xy, [243](#)
- set_bkg_tiles, [242](#)
- set_data, [248](#)
- set_default_palette, [247](#)
- set_interrupts, [244](#)
- set_native_sprite_data, [248](#)
- set_native_tile_data, [247](#)
- set_palette, [247](#)
- set_palette_entry, [247](#)
- SET_SHADOW_OAM_ADDRESS, [251](#)
- set_sprite_1bpp_data, [247](#)
- set_sprite_data, [248](#)
- set_sprite_palette, [242](#)
- set_sprite_palette_entry, [242](#)
- set_sprite_prop, [252](#)
- set_sprite_tile, [251](#)
- set_tile_1bpp_data, [248](#)
- set_tile_map, [249](#)
- set_tile_submap, [249](#)
- set_tile_submap_compat, [249](#)
- set_tile_xy, [254](#)
- set_vram_byte, [253](#)
- set_win_based_submap, [251](#)
- set_win_based_tiles, [249](#)
- set_win_submap, [250](#)
- set_win_tile_xy, [243](#)
- set_win_tiles, [242](#)
- shadow_OAM, [255](#)
- SHOW_BKG, [240](#)
- SHOW_LEFT_COLUMN, [240](#)
- SHOW_SPRITES, [240](#)
- SHOW_WIN, [240](#)
- SIO_IFLAG, [239](#)
- SPRITES_16x16, [240](#)
- SPRITES_8x8, [240](#)
- SWITCH_RAM, [242](#)
- SWITCH_ROM, [246](#)
- SWITCH_ROM1, [241](#)
- SWITCH_ROM2, [241](#)
- sys_time, [255](#)
- TIM_IFLAG, [238](#)
- VBK_REG, [237](#)
- VBL_IFLAG, [238](#)
- vmemcpy, [249](#)
- wait_vbl_done, [245](#)
- waitpad, [246](#)
- waitpadup, [246](#)
- WRITE_VDP_CMD, [243](#)
- WRITE_VDP_DATA, [243](#)
- NAKED
 - types.h, [89](#)
- nes.h
 - _current_1bpp_colors, [284](#)
 - _current_bank, [284](#)
 - _map_tile_offset, [284](#)
 - _shadow_OAM_base, [284](#)
 - _submap_tile_offset, [284](#)
 - BANK, [262](#)
 - BANKREF, [264](#)
 - BANKREF_EXTERN, [264](#)
 - COMPAT_PALETTE, [266](#)
 - CURRENT_BANK, [262](#)
 - delay, [267](#)
 - disable_interrupts, [269](#)
 - DISABLE_OAM_DMA, [266](#)
 - DISABLE_VBL_TRANSFER, [266](#)
 - DISPLAY_OFF, [265](#)

display_off, 270
DISPLAY_ON, 265
DMG_BLACK, 262
DMG_DARK_GRAY, 262
DMG_LITE_GRAY, 262
DMG_PALETTE, 262
DMG_WHITE, 262
enable_interrupts, 269
ENABLE_OAM_DMA, 266
ENABLE_VBL_TRANSFER, 266
fill_bkg_rect, 283
fill_rect, 266
get_bkg_tile_xy, 277
get_bkg_tiles, 276
get_bkg_xy_addr, 270
get_mode, 267
get_sprite_prop, 280
get_sprite_tile, 279
HIDE_BKG, 265
HIDE_LEFT_COLUMN, 265
hide_sprite, 281
HIDE_SPRITES, 265
init_bkg, 283
J_A, 260
J_B, 260
J_DOWN, 260
J_LEFT, 260
J_RIGHT, 260
J_SELECT, 260
J_START, 260
J_UP, 260
joypad, 267
joypad_ex, 269
joypad_init, 268
M_DRAWING, 260
M_NO_INTERP, 261
M_NO_SCROLL, 261
M_TEXT_INOUT, 261
M_TEXT_OUT, 261
MAX_HARDWARE_SPRITES, 266
mode, 267
move_bkg, 277
move_sprite, 280
NINTENDO_ENTERTAINMENT_SYSTEM, 259
OAM_item_t, 266
palette_color_t, 266
refresh_OAM, 270
RGB, 259
RGB8, 259
RGB_AQUA, 259
RGB_BLACK, 260
RGB_BLUE, 259
RGB_CYAN, 259
RGB_DARKBLUE, 259
RGB_DARKGRAY, 260
RGB_DARKGREEN, 259
RGB_DARKRED, 259
RGB_DARKYELLOW, 259
RGB_GREEN, 259
RGB_LIGHTGRAY, 260
RGB_PINK, 260
RGB_PURPLE, 260
RGB_RED, 259
RGB_WHITE, 260
RGB_YELLOW, 259
RGBHTML, 259
S_FLIPX, 261
S_FLIPY, 261
S_PALETTE, 261
S_PRIORITY, 261
SCREENHEIGHT, 262
SCREENWIDTH, 262
scroll_bkg, 277
scroll_sprite, 281
set_1bpp_colors, 271
set_1bpp_colors_ex, 271
set_2bpp_palette, 270
set_bkg_1bpp_data, 271
set_bkg_2bpp_data, 266
set_bkg_attributes, 273
set_bkg_based_submap, 275
set_bkg_based_tiles, 273
set_bkg_data, 271
set_bkg_palette, 267
set_bkg_palette_entry, 267
set_bkg_submap, 274
set_bkg_tile_xy, 276
set_bkg_tiles, 272
set_data, 281
set_native_tile_data, 282
SET_SHADOW_OAM_ADDRESS, 279
set_sprite_1bpp_data, 278
set_sprite_2bpp_data, 266
set_sprite_data, 278
set_sprite_palette, 267
set_sprite_palette_entry, 267
set_sprite_prop, 279
set_sprite_tile, 279
set_tile_data, 282
set_tile_map, 266
set_tile_submap, 266
set_tile_xy, 266
set_tiles, 282
set_vram_byte, 270
shadow_OAM, 284
SHOW_BKG, 265
SHOW_LEFT_COLUMN, 265
SHOW_SPRITES, 265
SPRITES_8x16, 265
SPRITES_8x8, 266
SWITCH_ROM, 264
SWITCH_ROM_DUMMY, 264
sys_time, 284
vmemset, 283
wait_vbl_done, 270
waitpad, 268

waitpadup, [268](#)
 NINTENDO
 gb.h, [114](#)
 NINTENDO_ENTERTAINMENT_SYSTEM
 nes.h, [259](#)
 NONBANKED
 types.h, [89](#)
 noreturn
 stdnoreturn.h, [323](#)
 nowait_int_handler
 gb.h, [127](#)
 npads
 joypads_t, [68](#)
 NR10_REG
 hardware.h, [176](#)
 NR11_REG
 hardware.h, [176](#)
 NR12_REG
 hardware.h, [176](#)
 NR13_REG
 hardware.h, [177](#)
 NR14_REG
 hardware.h, [177](#)
 NR21_REG
 hardware.h, [177](#)
 NR22_REG
 hardware.h, [177](#)
 NR23_REG
 hardware.h, [177](#)
 NR24_REG
 hardware.h, [177](#)
 NR30_REG
 hardware.h, [177](#)
 NR31_REG
 hardware.h, [177](#)
 NR32_REG
 hardware.h, [177](#)
 NR33_REG
 hardware.h, [177](#)
 NR34_REG
 hardware.h, [177](#)
 NR41_REG
 hardware.h, [177](#)
 NR42_REG
 hardware.h, [177](#)
 NR43_REG
 hardware.h, [177](#)
 NR44_REG
 hardware.h, [177](#)
 NR50_REG
 hardware.h, [177](#)
 NR51_REG
 hardware.h, [177](#)
 NR52_REG
 hardware.h, [177](#)
 NULL
 stddef.h, [311](#)
 types.h, [92](#)
 OAM_item_t, [70](#)
 gb.h, [124](#)
 msx.h, [243](#)
 nes.h, [266](#)
 prop, [71](#)
 tile, [71](#)
 x, [71](#)
 y, [71](#)
 OAMADDR
 hardware.h, [191](#)
 OAMDATA
 hardware.h, [191](#)
 OAMDMA
 hardware.h, [191](#)
 OAMF_BANK0
 hardware.h, [174](#)
 OAMF_BANK1
 hardware.h, [174](#)
 OAMF_CGB_PAL0
 hardware.h, [174](#)
 OAMF_CGB_PAL1
 hardware.h, [174](#)
 OAMF_CGB_PAL2
 hardware.h, [174](#)
 OAMF_CGB_PAL3
 hardware.h, [174](#)
 OAMF_CGB_PAL4
 hardware.h, [174](#)
 OAMF_CGB_PAL5
 hardware.h, [174](#)
 OAMF_CGB_PAL6
 hardware.h, [174](#)
 OAMF_CGB_PAL7
 hardware.h, [174](#)
 OAMF_PAL0
 hardware.h, [173](#)
 OAMF_PAL1
 hardware.h, [173](#)
 OAMF_PALMASK
 hardware.h, [174](#)
 OAMF_PRI
 hardware.h, [173](#)
 OAMF_XFLIP
 hardware.h, [173](#)
 OAMF_YFLIP
 hardware.h, [173](#)
 OBP0_REG
 hardware.h, [178](#)
 OBP1_REG
 hardware.h, [178](#)
 OCPD_REG
 hardware.h, [179](#)
 OCPS_REG
 hardware.h, [179](#)
 OCPSF_AUTOINC
 hardware.h, [172](#)
 offsetof
 stddef.h, [311](#)

- ofs
 - [__far_ptr](#), 65
- OLDCALL
 - [types.h](#), 89
- opcode
 - [isr_nested_vector_t](#), 66
 - [isr_vector_t](#), 67
- OR
 - [drawing.h](#), 103
- P1_REG
 - [hardware.h](#), 176
- P1F_0
 - [hardware.h](#), 163
- P1F_1
 - [hardware.h](#), 163
- P1F_2
 - [hardware.h](#), 163
- P1F_3
 - [hardware.h](#), 163
- P1F_4
 - [hardware.h](#), 163
- P1F_5
 - [hardware.h](#), 163
- P1F_GET_BTN
 - [hardware.h](#), 163
- P1F_GET_DPAD
 - [hardware.h](#), 163
- P1F_GET_NONE
 - [hardware.h](#), 163
- palette_color_t
 - [cgb.h](#), 99
 - [nes.h](#), 266
- PCM12_REG
 - [hardware.h](#), 179
- PCM34_REG
 - [hardware.h](#), 179
- PCM_SAMPLE
 - [hardware.h](#), 178
- plot
 - [drawing.h](#), 105
- plot_point
 - [drawing.h](#), 105
- pmfont_handle
 - [font.h](#), 228
- POINTER
 - [types.h](#), 92
- posx
 - [console.h](#), 223
- posy
 - [console.h](#), 223
- PPUADDR
 - [hardware.h](#), 191
- PPUCTRL
 - [hardware.h](#), 190
- PPUCTRL_BG_CHR
 - [hardware.h](#), 190
- PPUCTRL_INC32
 - [hardware.h](#), 190
- PPUCTRL_NMI
 - [hardware.h](#), 190
- PPUCTRL_SPR_8X16
 - [hardware.h](#), 190
- PPUCTRL_SPR_8X8
 - [hardware.h](#), 190
- PPUCTRL_SPR_CHR
 - [hardware.h](#), 190
- PPUDATA
 - [hardware.h](#), 191
- PPUMASK
 - [hardware.h](#), 190
- PPUMASK_BLUE
 - [hardware.h](#), 190
- PPUMASK_GREEN
 - [hardware.h](#), 190
- PPUMASK_MONOCHROME
 - [hardware.h](#), 191
- PPUMASK_RED
 - [hardware.h](#), 190
- PPUMASK_SHOW_BG
 - [hardware.h](#), 191
- PPUMASK_SHOW_BG_LC
 - [hardware.h](#), 191
- PPUMASK_SHOW_SPR
 - [hardware.h](#), 190
- PPUMASK_SHOW_SPR_LC
 - [hardware.h](#), 191
- PPUSCROLL
 - [hardware.h](#), 191
- PPUSTATUS
 - [hardware.h](#), 191
- PRESERVES_REGS
 - [types.h](#), 89
- printf
 - [stdio.h](#), 318
- prop
 - [OAM_item_t](#), 71
- props
 - [metasprite_t](#), 70
- provides.h
 - [USE_C_MEMCPY](#), 72, 73
 - [USE_C_STRCMP](#), 72, 73
 - [USE_C_STRCPY](#), 72, 73
- PSG_CH0
 - [hardware.h](#), 183, 196
- PSG_CH1
 - [hardware.h](#), 183, 196
- PSG_CH2
 - [hardware.h](#), 183, 196
- PSG_CH3
 - [hardware.h](#), 183, 196
- PSG_LATCH
 - [hardware.h](#), 183, 196
- PSG_VOLUME
 - [hardware.h](#), 183, 196
- ptr
 - [__far_ptr](#), 65

PTRDIFF_MAX
 stdint.h, 315
 PTRDIFF_MIN
 stdint.h, 315
 ptrdiff_t
 stddef.h, 312
 putchar
 stdio.h, 318
 puts
 stdio.h, 319

 qsort
 stdlib.h, 323

 R0_CB_INPUT
 hardware.h, 183
 R0_CB_OUTPUT
 hardware.h, 183
 R0_DEFAULT
 hardware.h, 183, 197
 R0_ES
 hardware.h, 184, 197
 R0_ES_OFF
 hardware.h, 184, 197
 R0_HSCRL
 hardware.h, 196
 R0_HSCRL_INH
 hardware.h, 196
 R0_IE1
 hardware.h, 184, 197
 R0_IE1_OFF
 hardware.h, 184, 197
 R0_IE2
 hardware.h, 184
 R0_IE2_OFF
 hardware.h, 183
 R0_LCB
 hardware.h, 196
 R0_NO_LCB
 hardware.h, 196
 R0_SCR_MODE1
 hardware.h, 184
 R0_SCR_MODE2
 hardware.h, 184
 R0_SCR_MODE3
 hardware.h, 184
 R0_SS
 hardware.h, 197
 R0_SS_OFF
 hardware.h, 197
 R0_VSCRL
 hardware.h, 196
 R0_VSCRL_INH
 hardware.h, 196
 R10_INT_EVERY
 hardware.h, 186, 199
 R10_INT_OFF
 hardware.h, 186, 199
 R1_DEFAULT
 hardware.h, 184, 197
 R1_DISP_OFF
 hardware.h, 184, 197
 R1_DISP_ON
 hardware.h, 184, 197
 R1_IE
 hardware.h, 184, 197
 R1_IE_OFF
 hardware.h, 184, 197
 R1_SCR_MODE1
 hardware.h, 184
 R1_SCR_MODE2
 hardware.h, 184
 R1_SCR_MODE3
 hardware.h, 184
 R1_SPR_16X16
 hardware.h, 185
 R1_SPR_8X16
 hardware.h, 197
 R1_SPR_8X8
 hardware.h, 184, 197
 R1_SPR_MAG
 hardware.h, 185
 R1_SPR_MAG_OFF
 hardware.h, 185
 R2_MAP_0x0000
 hardware.h, 185, 198
 R2_MAP_0x0800
 hardware.h, 185, 198
 R2_MAP_0x1000
 hardware.h, 185, 198
 R2_MAP_0x1800
 hardware.h, 185, 198
 R2_MAP_0x2000
 hardware.h, 185, 198
 R2_MAP_0x2800
 hardware.h, 185, 198
 R2_MAP_0x3000
 hardware.h, 185, 197
 R2_MAP_0x3800
 hardware.h, 185, 197
 R5_SAT_0x3F00
 hardware.h, 185, 198
 R5_SAT_MASK
 hardware.h, 185, 198
 R6_BANK0
 hardware.h, 186, 198
 R6_BANK1
 hardware.h, 186, 198
 R6_DATA_0x0000
 hardware.h, 186, 198
 R6_DATA_0x2000
 hardware.h, 186, 198
 R7_COLOR_MASK
 hardware.h, 186, 199
 RAMCTL_BANK
 hardware.h, 200
 RAMCTL_PROT

- hardware.h, [200](#)
- RAMCTL_RAM
 - hardware.h, [200](#)
- RAMCTL_RO
 - hardware.h, [200](#)
- RAMCTL_ROM
 - hardware.h, [200](#)
- rand
 - rand.h, [285](#)
- rand.h
 - __rand_seed, [286](#)
 - arand, [286](#)
 - initarand, [286](#)
 - initrand, [285](#)
 - rand, [285](#)
 - RAND_MAX, [285](#)
 - randw, [285](#)
 - RANDW_MAX, [285](#)
- RAND_MAX
 - rand.h, [285](#)
- randw
 - rand.h, [285](#)
- RANDW_MAX
 - rand.h, [285](#)
- rAUD1ENV
 - hardware.h, [165](#)
- rAUD1HIGH
 - hardware.h, [165](#)
- rAUD1LEN
 - hardware.h, [165](#)
- rAUD1LOW
 - hardware.h, [165](#)
- rAUD1SWEEP
 - hardware.h, [164](#)
- rAUD2ENV
 - hardware.h, [165](#)
- rAUD2HIGH
 - hardware.h, [165](#)
- rAUD2LEN
 - hardware.h, [165](#)
- rAUD2LOW
 - hardware.h, [165](#)
- rAUD3ENA
 - hardware.h, [165](#)
- rAUD3HIGH
 - hardware.h, [165](#)
- rAUD3LEN
 - hardware.h, [165](#)
- rAUD3LEVEL
 - hardware.h, [165](#)
- rAUD3LOW
 - hardware.h, [165](#)
- rAUD4ENV
 - hardware.h, [165](#)
- rAUD4GO
 - hardware.h, [166](#)
- rAUD4LEN
 - hardware.h, [165](#)
- rAUD4POLY
 - hardware.h, [165](#)
- rAUDENA
 - hardware.h, [167](#)
- rAUDTERM
 - hardware.h, [166](#)
- rAUDVOL
 - hardware.h, [166](#)
- rBCPD
 - hardware.h, [172](#)
- rBCPS
 - hardware.h, [172](#)
- rBGP
 - hardware.h, [169](#)
- rDIV
 - hardware.h, [163](#)
- rDMA
 - hardware.h, [169](#)
- realloc
 - stdlib.h, [322](#)
- receive_byte
 - gb.h, [128](#)
- refresh_OAM
 - gb.h, [131](#)
 - msx.h, [246](#)
 - nes.h, [270](#)
 - sms.h, [300](#)
- remove_JOY
 - gb.h, [125](#)
 - msx.h, [245](#)
 - sms.h, [299](#)
- remove_LCD
 - gb.h, [125](#)
 - msx.h, [244](#)
 - sms.h, [299](#)
- remove_SIO
 - gb.h, [125](#)
 - msx.h, [245](#)
 - sms.h, [299](#)
- remove_TIM
 - gb.h, [125](#)
 - msx.h, [244](#)
 - sms.h, [299](#)
- remove_VBL
 - gb.h, [124](#)
 - msx.h, [244](#)
 - sms.h, [298](#)
- reset
 - gb.h, [130](#)
- RET_SIZE
 - setjmp.h, [287](#)
- reverse
 - string.h, [77](#), [80](#), [84](#)
- RGB
 - cgb.h, [97](#)
 - nes.h, [259](#)
- RGB8
 - cgb.h, [98](#)

nes.h, [259](#)
 RGB_AQUA
 cgb.h, [99](#)
 nes.h, [259](#)
 RGB_BLACK
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_BLUE
 cgb.h, [98](#)
 nes.h, [259](#)
 RGB_BROWN
 cgb.h, [99](#)
 RGB_CYAN
 cgb.h, [99](#)
 nes.h, [259](#)
 RGB_DARKBLUE
 cgb.h, [99](#)
 nes.h, [259](#)
 RGB_DARKGRAY
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_DARKGREEN
 cgb.h, [98](#)
 nes.h, [259](#)
 RGB_DARKRED
 cgb.h, [98](#)
 nes.h, [259](#)
 RGB_DARKYELLOW
 cgb.h, [99](#)
 nes.h, [259](#)
 RGB_GREEN
 cgb.h, [98](#)
 nes.h, [259](#)
 RGB_LIGHTFLESH
 cgb.h, [99](#)
 RGB_LIGHTGRAY
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_ORANGE
 cgb.h, [99](#)
 RGB_PINK
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_PURPLE
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_RED
 cgb.h, [98](#)
 nes.h, [259](#)
 RGB_TEAL
 cgb.h, [99](#)
 RGB_TO_NES
 rgb_to_nes_macro.h, [284](#)
 rgb_to_nes_macro.h
 RGB_TO_NES, [284](#)
 RGB_WHITE
 cgb.h, [99](#)
 nes.h, [260](#)
 RGB_YELLOW
 cgb.h, [99](#)
 nes.h, [259](#)
 RGBHTML
 cgb.h, [98](#)
 nes.h, [259](#)
 rHDMA1
 hardware.h, [171](#)
 rHDMA2
 hardware.h, [171](#)
 rHDMA3
 hardware.h, [171](#)
 rHDMA4
 hardware.h, [171](#)
 rHDMA5
 hardware.h, [171](#)
 rIE
 hardware.h, [172](#)
 rIF
 hardware.h, [164](#)
 rKEY1
 hardware.h, [170](#)
 rLDC
 hardware.h, [167](#)
 rle_decompress
 rledecompress.h, [231](#)
 rle_init
 rledecompress.h, [231](#)
 RLE_STOP
 rledecompress.h, [231](#)
 rledecompress.h
 rle_decompress, [231](#)
 rle_init, [231](#)
 RLE_STOP, [231](#)
 rLY
 hardware.h, [169](#)
 rLYC
 hardware.h, [169](#)
 rOBP0
 hardware.h, [169](#)
 rOBP1
 hardware.h, [170](#)
 rOCPD
 hardware.h, [172](#)
 rOCP5
 hardware.h, [172](#)
 rP1
 hardware.h, [163](#)
 RP_REG
 hardware.h, [179](#)
 rPCM12
 hardware.h, [172](#)
 rPCM34
 hardware.h, [172](#)
 RPF_DATAIN
 hardware.h, [171](#)
 RPF_ENREAD
 hardware.h, [171](#)

RPF_WRITE_HI
 hardware.h, 172

RPF_WRITE_LO
 hardware.h, 172

rRAMB
 hardware.h, 176

rRAMG
 hardware.h, 176

rROMB0
 hardware.h, 176

rROMB1
 hardware.h, 176

rRP
 hardware.h, 171

rSB
 hardware.h, 163

rSC
 hardware.h, 163

rSCX
 hardware.h, 169

rSCY
 hardware.h, 169

rSMBK
 hardware.h, 172

rSPD
 hardware.h, 170

rSTAT
 hardware.h, 168

rSVBK
 hardware.h, 172

rTAC
 hardware.h, 163

rTIMA
 hardware.h, 163

rTMA
 hardware.h, 163

rVBK
 hardware.h, 170

rWX
 hardware.h, 170

rWY
 hardware.h, 170

S_FLIPX
 gb.h, 115
 msx.h, 238
 nes.h, 261
 sms.h, 292

S_FLIPY
 gb.h, 115
 msx.h, 238
 nes.h, 261
 sms.h, 292

S_PALETTE
 gb.h, 115
 msx.h, 238
 nes.h, 261
 sms.h, 292

S_PRIORITY
 gb.h, 115
 msx.h, 238
 nes.h, 261
 sms.h, 292

SB_REG
 hardware.h, 176

SC_REG
 hardware.h, 176

SCHAR_MAX
 limits.h, 232

SCHAR_MIN
 limits.h, 232

SCREENHEIGHT
 gb.h, 117
 msx.h, 239
 nes.h, 262
 sms.h, 293

SCREENWIDTH
 gb.h, 117
 msx.h, 239
 nes.h, 262
 sms.h, 293

scroll_bkg
 gb.h, 138
 msx.h, 245
 nes.h, 277
 sms.h, 299

scroll_sprite
 gb.h, 146
 msx.h, 253
 nes.h, 281
 sms.h, 307

scroll_win
 gb.h, 143

SCX_REG
 hardware.h, 178

SCY_REG
 hardware.h, 178

seg
 __far_ptr, 65

SEGA
 sms.h, 291

segfn
 __far_ptr, 65

segofs
 __far_ptr, 65

send_byte
 gb.h, 128

set_1bpp_colors
 gb.h, 132
 msx.h, 248
 nes.h, 271
 sms.h, 302

set_1bpp_colors_ex
 gb.h, 131
 nes.h, 271

set_2bpp_palette
 gb.h, 131

- msx.h, [248](#)
- nes.h, [270](#)
- sms.h, [302](#)
- set_attributed_tile_xy
 - msx.h, [253](#)
 - sms.h, [308](#)
- set_bkg_1bpp_data
 - gb.h, [133](#)
 - msx.h, [248](#)
 - nes.h, [271](#)
 - sms.h, [302](#)
- set_bkg_2bpp_data
 - gb.h, [123](#)
 - nes.h, [266](#)
 - sms.h, [302](#)
- set_bkg_4bpp_data
 - msx.h, [247](#)
 - sms.h, [301](#)
- set_bkg_attributes
 - nes.h, [273](#)
- set_bkg_based_submap
 - gb.h, [136](#)
 - msx.h, [251](#)
 - nes.h, [275](#)
 - sms.h, [305](#)
- set_bkg_based_tiles
 - gb.h, [134](#)
 - msx.h, [249](#)
 - nes.h, [273](#)
 - sms.h, [303](#)
- set_bkg_data
 - gb.h, [132](#)
 - msx.h, [248](#)
 - nes.h, [271](#)
 - sms.h, [302](#)
- set_bkg_palette
 - cgb.h, [100](#)
 - msx.h, [242](#)
 - nes.h, [267](#)
 - sms.h, [296](#)
- set_bkg_palette_entry
 - cgb.h, [100](#)
 - msx.h, [242](#)
 - nes.h, [267](#)
 - sms.h, [296](#)
- set_bkg_submap
 - gb.h, [135](#)
 - msx.h, [249](#)
 - nes.h, [274](#)
 - sms.h, [304](#)
- set_bkg_tile_xy
 - gb.h, [137](#)
 - msx.h, [243](#)
 - nes.h, [276](#)
 - sms.h, [297](#)
- set_bkg_tiles
 - gb.h, [133](#)
 - msx.h, [242](#)
- nes.h, [272](#)
- sms.h, [297](#)
- set_data
 - gb.h, [147](#)
 - msx.h, [248](#)
 - nes.h, [281](#)
 - sms.h, [303](#)
- set_default_palette
 - cgb.h, [102](#)
 - msx.h, [247](#)
 - sms.h, [301](#)
- set_interrupts
 - gb.h, [130](#)
 - msx.h, [244](#)
 - sms.h, [298](#)
- set_native_sprite_data
 - msx.h, [248](#)
- set_native_tile_data
 - gb.h, [150](#)
 - msx.h, [247](#)
 - nes.h, [282](#)
 - sms.h, [301](#)
- set_palette
 - msx.h, [247](#)
 - sms.h, [301](#)
- set_palette_entry
 - msx.h, [247](#)
 - sms.h, [301](#)
- SET_SHADOW_OAM_ADDRESS
 - gb.h, [145](#)
 - msx.h, [251](#)
 - nes.h, [279](#)
 - sms.h, [306](#)
- set_sprite_1bpp_data
 - gb.h, [144](#)
 - msx.h, [247](#)
 - nes.h, [278](#)
 - sms.h, [303](#)
- set_sprite_2bpp_data
 - gb.h, [123](#)
 - nes.h, [266](#)
 - sms.h, [302](#)
- set_sprite_4bpp_data
 - sms.h, [301](#)
- set_sprite_data
 - gb.h, [143](#)
 - msx.h, [248](#)
 - nes.h, [278](#)
 - sms.h, [302](#)
- set_sprite_palette
 - cgb.h, [100](#)
 - msx.h, [242](#)
 - nes.h, [267](#)
 - sms.h, [297](#)
- set_sprite_palette_entry
 - cgb.h, [101](#)
 - msx.h, [242](#)
 - nes.h, [267](#)

- sms.h, [296](#)
- set_sprite_prop
 - gb.h, [145](#)
 - msx.h, [252](#)
 - nes.h, [279](#)
 - sms.h, [307](#)
- set_sprite_tile
 - gb.h, [145](#)
 - msx.h, [251](#)
 - nes.h, [279](#)
 - sms.h, [306](#)
- set_tile_1bpp_data
 - msx.h, [248](#)
 - sms.h, [302](#)
- set_tile_2bpp_data
 - sms.h, [302](#)
- set_tile_data
 - gb.h, [149](#)
 - nes.h, [282](#)
- set_tile_map
 - gb.h, [123](#)
 - msx.h, [249](#)
 - nes.h, [266](#)
 - sms.h, [303](#)
- set_tile_map_compat
 - sms.h, [303](#)
- set_tile_submap
 - gb.h, [123](#)
 - msx.h, [249](#)
 - nes.h, [266](#)
 - sms.h, [304](#)
- set_tile_submap_compat
 - msx.h, [249](#)
 - sms.h, [304](#)
- set_tile_xy
 - gb.h, [123](#)
 - msx.h, [254](#)
 - nes.h, [266](#)
 - sms.h, [308](#)
- set_tiles
 - gb.h, [148](#)
 - nes.h, [282](#)
- set_vram_byte
 - gb.h, [131](#)
 - msx.h, [253](#)
 - nes.h, [270](#)
 - sms.h, [308](#)
- set_win_1bpp_data
 - gb.h, [139](#)
- set_win_based_submap
 - gb.h, [141](#)
 - msx.h, [251](#)
 - sms.h, [306](#)
- set_win_based_tiles
 - gb.h, [140](#)
 - msx.h, [249](#)
 - sms.h, [303](#)
- set_win_data
 - gb.h, [138](#)
- set_win_submap
 - gb.h, [140](#)
 - msx.h, [250](#)
 - sms.h, [305](#)
- set_win_tile_xy
 - gb.h, [142](#)
 - msx.h, [243](#)
 - sms.h, [297](#)
- set_win_tiles
 - gb.h, [139](#)
 - msx.h, [242](#)
 - sms.h, [297](#)
- setchar
 - console.h, [223](#)
- setjmp
 - setjmp.h, [287](#)
- setjmp.h
 - __setjmp, [287](#)
 - BP_SIZE, [287](#)
 - BPX_SIZE, [287](#)
 - jmp_buf, [287](#)
 - longjmp, [287](#)
 - RET_SIZE, [287](#)
 - setjmp, [287](#)
 - SP_SIZE, [286](#)
 - SPX_SIZE, [287](#)
- sfont_handle, [71](#)
 - first_tile, [71](#)
 - font, [71](#)
- SFR
 - types.h, [89](#)
- sgb.h
 - c, [223](#)
 - SGB_ATTRC_EN, [221](#)
 - SGB_ATTR_BLK, [221](#)
 - SGB_ATTR_CHR, [221](#)
 - SGB_ATTR_DIV, [221](#)
 - SGB_ATTR_LIN, [221](#)
 - SGB_ATTR_SET, [222](#)
 - SGB_ATTR_TRN, [222](#)
 - sgb_check, [222](#)
 - SGB_CHR_TRN, [222](#)
 - SGB_DATA_SND, [221](#)
 - SGB_DATA_TRN, [222](#)
 - SGB_ICON_EN, [221](#)
 - SGB_JUMP, [222](#)
 - SGB_MASK_EN, [222](#)
 - SGB_MLT_REQ, [222](#)
 - SGB_OBJ_TRN, [222](#)
 - SGB_PAL_01, [221](#)
 - SGB_PAL_03, [221](#)
 - SGB_PAL_12, [221](#)
 - SGB_PAL_23, [221](#)
 - SGB_PAL_SET, [221](#)
 - SGB_PAL_TRN, [221](#)
 - SGB_PCT_TRN, [222](#)
 - SGB_SOU_TRN, [221](#)

SGB_SOUND, [221](#)
SGB_TEST_EN, [221](#)
sgb_transfer, [222](#)
SGB_ATTRC_EN
 sgb.h, [221](#)
SGB_ATTR_BLK
 sgb.h, [221](#)
SGB_ATTR_CHR
 sgb.h, [221](#)
SGB_ATTR_DIV
 sgb.h, [221](#)
SGB_ATTR_LIN
 sgb.h, [221](#)
SGB_ATTR_SET
 sgb.h, [222](#)
SGB_ATTR_TRN
 sgb.h, [222](#)
sgb_check
 sgb.h, [222](#)
SGB_CHR_TRN
 sgb.h, [222](#)
SGB_DATA_SND
 sgb.h, [221](#)
SGB_DATA_TRN
 sgb.h, [222](#)
SGB_ICON_EN
 sgb.h, [221](#)
SGB_JUMP
 sgb.h, [222](#)
SGB_MASK_EN
 sgb.h, [222](#)
SGB_MLT_REQ
 sgb.h, [222](#)
SGB_OBJ_TRN
 sgb.h, [222](#)
SGB_PAL_01
 sgb.h, [221](#)
SGB_PAL_03
 sgb.h, [221](#)
SGB_PAL_12
 sgb.h, [221](#)
SGB_PAL_23
 sgb.h, [221](#)
SGB_PAL_SET
 sgb.h, [221](#)
SGB_PAL_TRN
 sgb.h, [221](#)
SGB_PCT_TRN
 sgb.h, [222](#)
SGB_SOU_TRN
 sgb.h, [221](#)
SGB_SOUND
 sgb.h, [221](#)
SGB_TEST_EN
 sgb.h, [221](#)
sgb_transfer
 sgb.h, [222](#)
shadow_OAM
 gb.h, [153](#)
 msx.h, [255](#)
 nes.h, [284](#)
 sms.h, [310](#)
shadow_PPUCTRL
 hardware.h, [192](#)
shadow_PPUMASK
 hardware.h, [192](#)
shadow_VDP_R0
 hardware.h, [188](#), [201](#)
shadow_VDP_R1
 hardware.h, [188](#), [201](#)
shadow_VDP_R10
 hardware.h, [188](#), [201](#)
shadow_VDP_R2
 hardware.h, [188](#), [201](#)
shadow_VDP_R3
 hardware.h, [188](#), [201](#)
shadow_VDP_R4
 hardware.h, [188](#), [201](#)
shadow_VDP_R5
 hardware.h, [188](#), [201](#)
shadow_VDP_R6
 hardware.h, [188](#), [201](#)
shadow_VDP_R7
 hardware.h, [188](#), [201](#)
shadow_VDP_R8
 hardware.h, [188](#), [201](#)
shadow_VDP_R9
 hardware.h, [188](#), [201](#)
shadow_VDP_RBORDER
 hardware.h, [188](#), [201](#)
shadow_VDP_RSCX
 hardware.h, [188](#), [201](#)
shadow_VDP_RSCY
 hardware.h, [188](#), [201](#)
SHOW_BKG
 gb.h, [123](#)
 msx.h, [240](#)
 nes.h, [265](#)
 sms.h, [294](#)
SHOW_LEFT_COLUMN
 gb.h, [122](#)
 msx.h, [240](#)
 nes.h, [265](#)
 sms.h, [294](#)
SHOW_SPRITES
 gb.h, [123](#)
 msx.h, [240](#)
 nes.h, [265](#)
 sms.h, [294](#)
SHOW_WIN
 gb.h, [123](#)
 msx.h, [240](#)
 sms.h, [294](#)
SHRT_MAX
 limits.h, [232](#)
SHRT_MIN

- limits.h, [233](#)
- SIG_ATOMIC_MAX
 - stdint.h, [315](#)
- SIG_ATOMIC_MIN
 - stdint.h, [315](#)
- SIGNED
 - drawing.h, [104](#)
- SIO_IFLAG
 - gb.h, [116](#)
 - msx.h, [239](#)
 - sms.h, [293](#)
- SIOF_B_CLOCK
 - hardware.h, [164](#)
- SIOF_B_SPEED
 - hardware.h, [164](#)
- SIOF_B_XFER_START
 - hardware.h, [164](#)
- SIOF_CLOCK_EXT
 - hardware.h, [164](#)
- SIOF_CLOCK_INT
 - hardware.h, [164](#)
- SIOF_SPEED_1X
 - hardware.h, [164](#)
- SIOF_SPEED_32X
 - hardware.h, [164](#)
- SIOF_XFER_START
 - hardware.h, [164](#)
- SIZE_MAX
 - stdint.h, [315](#)
- size_t
 - stddef.h, [312](#)
 - types.h, [87](#), [88](#), [91](#)
- sms.h
 - __READ_VDP_REG, [292](#)
 - __WRITE_VDP_REG, [292](#)
 - _current_1bpp_colors, [309](#)
 - _current_2bpp_palette, [309](#)
 - _current_bank, [295](#)
 - _map_tile_offset, [309](#)
 - _shadow_OAM_OFF, [310](#)
 - _shadow_OAM_base, [310](#)
 - _submap_tile_offset, [309](#)
 - add_JOY, [299](#)
 - add_LCD, [299](#)
 - add_SIO, [299](#)
 - add_TIM, [299](#)
 - add_VBL, [299](#)
 - BANK, [295](#)
 - BANKREF, [295](#)
 - BANKREF_EXTERN, [295](#)
 - c, [309](#)
 - cancel_pending_interrupts, [299](#)
 - cgb_compatibility, [301](#)
 - COMPAT_PALETTE, [297](#)
 - cpu_fast, [301](#)
 - CURRENT_BANK, [295](#)
 - d, [309](#)
 - delay, [300](#)
 - DEVICE_SUPPORTS_COLOR, [295](#)
 - DISABLE_RAM, [296](#)
 - DISABLE_VBL_TRANSFER, [297](#)
 - DISPLAY_OFF, [294](#)
 - display_off, [300](#)
 - DISPLAY_ON, [293](#)
 - e, [309](#)
 - EMPTY_IFLAG, [292](#)
 - ENABLE_RAM, [296](#)
 - ENABLE_VBL_TRANSFER, [297](#)
 - fill_bkg_rect, [297](#)
 - fill_rect, [306](#)
 - fill_rect_compat, [306](#)
 - fill_win_rect, [297](#)
 - get_bkg_xy_addr, [309](#)
 - get_mode, [298](#)
 - get_sprite_prop, [307](#)
 - get_sprite_tile, [307](#)
 - get_win_xy_addr, [297](#)
 - h, [309](#)
 - HIDE_BKG, [294](#)
 - HIDE_LEFT_COLUMN, [294](#)
 - hide_sprite, [308](#)
 - HIDE_SPRITES, [294](#)
 - HIDE_WIN, [294](#)
 - int_handler, [297](#)
 - iyh, [309](#)
 - iyl, [309](#)
 - J_A, [291](#)
 - J_B, [291](#)
 - J_DOWN, [291](#)
 - J_LEFT, [291](#)
 - J_RIGHT, [291](#)
 - J_UP, [291](#)
 - JOY_IFLAG, [293](#)
 - joypad, [300](#)
 - joypad_ex, [300](#)
 - joypad_init, [300](#)
 - l, [309](#)
 - LCD_IFLAG, [292](#)
 - M_NO_INTERP, [292](#)
 - M_NO_SCROLL, [292](#)
 - M_TEXT_INOUT, [291](#)
 - M_TEXT_OUT, [291](#)
 - MAX_HARDWARE_SPRITES, [297](#)
 - MAXWNDPOSX, [293](#)
 - MAXWNDPOSY, [293](#)
 - MINWNDPOSX, [293](#)
 - MINWNDPOSY, [293](#)
 - mode, [298](#)
 - move_bkg, [299](#)
 - move_sprite, [307](#)
 - refresh_OAM, [300](#)
 - remove_JOY, [299](#)
 - remove_LCD, [299](#)
 - remove_SIO, [299](#)
 - remove_TIM, [299](#)
 - remove_VBL, [298](#)

[S_FLIPX](#), [292](#)
[S_FLIPY](#), [292](#)
[S_PALETTE](#), [292](#)
[S_PRIORITY](#), [292](#)
[SCREENHEIGHT](#), [293](#)
[SCREENWIDTH](#), [293](#)
[scroll_bkg](#), [299](#)
[scroll_sprite](#), [307](#)
[SEGA](#), [291](#)
[set_1bpp_colors](#), [302](#)
[set_2bpp_palette](#), [302](#)
[set_attributed_tile_xy](#), [308](#)
[set_bkg_1bpp_data](#), [302](#)
[set_bkg_2bpp_data](#), [302](#)
[set_bkg_4bpp_data](#), [301](#)
[set_bkg_based_submap](#), [305](#)
[set_bkg_based_tiles](#), [303](#)
[set_bkg_data](#), [302](#)
[set_bkg_palette](#), [296](#)
[set_bkg_palette_entry](#), [296](#)
[set_bkg_submap](#), [304](#)
[set_bkg_tile_xy](#), [297](#)
[set_bkg_tiles](#), [297](#)
[set_data](#), [303](#)
[set_default_palette](#), [301](#)
[set_interrupts](#), [298](#)
[set_native_tile_data](#), [301](#)
[set_palette](#), [301](#)
[set_palette_entry](#), [301](#)
[SET_SHADOW_OAM_ADDRESS](#), [306](#)
[set_sprite_1bpp_data](#), [303](#)
[set_sprite_2bpp_data](#), [302](#)
[set_sprite_4bpp_data](#), [301](#)
[set_sprite_data](#), [302](#)
[set_sprite_palette](#), [297](#)
[set_sprite_palette_entry](#), [296](#)
[set_sprite_prop](#), [307](#)
[set_sprite_tile](#), [306](#)
[set_tile_1bpp_data](#), [302](#)
[set_tile_2bpp_data](#), [302](#)
[set_tile_map](#), [303](#)
[set_tile_map_compat](#), [303](#)
[set_tile_submap](#), [304](#)
[set_tile_submap_compat](#), [304](#)
[set_tile_xy](#), [308](#)
[set_vram_byte](#), [308](#)
[set_win_based_submap](#), [306](#)
[set_win_based_tiles](#), [303](#)
[set_win_submap](#), [305](#)
[set_win_tile_xy](#), [297](#)
[set_win_tiles](#), [297](#)
[shadow_OAM](#), [310](#)
[SHOW_BKG](#), [294](#)
[SHOW_LEFT_COLUMN](#), [294](#)
[SHOW_SPRITES](#), [294](#)
[SHOW_WIN](#), [294](#)
[SIO_IFLAG](#), [293](#)
[SPRITES_8x16](#), [294](#)
[SPRITES_8x8](#), [294](#)
[SWITCH_RAM](#), [296](#)
[SWITCH_ROM](#), [296](#)
[SWITCH_ROM1](#), [296](#)
[SWITCH_ROM2](#), [296](#)
[sys_time](#), [309](#)
[TIM_IFLAG](#), [293](#)
[VBK_REG](#), [291](#)
[VBL_IFLAG](#), [292](#)
[vmemcpy](#), [303](#)
[wait_vbl_done](#), [300](#)
[waitpad](#), [300](#)
[waitpadup](#), [300](#)
[WRITE_VDP_CMD](#), [298](#)
[WRITE_VDP_DATA](#), [298](#)
[SOLID](#)
 [drawing.h](#), [103](#)
[SP_SIZE](#)
 [setjmp.h](#), [286](#)
[sprintf](#)
 [stdio.h](#), [319](#)
[SPRITES_16x16](#)
 [msx.h](#), [240](#)
[SPRITES_8x16](#)
 [gb.h](#), [123](#)
 [nes.h](#), [265](#)
 [sms.h](#), [294](#)
[SPRITES_8x8](#)
 [gb.h](#), [123](#)
 [msx.h](#), [240](#)
 [nes.h](#), [266](#)
 [sms.h](#), [294](#)
[SPX_SIZE](#)
 [setjmp.h](#), [287](#)
[STAT_REG](#)
 [hardware.h](#), [178](#)
[STATF_9_SPR](#)
 [hardware.h](#), [183](#), [196](#)
[STATF_B_BUSY](#)
 [hardware.h](#), [169](#)
[STATF_B_LYC](#)
 [hardware.h](#), [169](#)
[STATF_B_LYCF](#)
 [hardware.h](#), [169](#)
[STATF_B_MODE00](#)
 [hardware.h](#), [169](#)
[STATF_B_MODE01](#)
 [hardware.h](#), [169](#)
[STATF_B_MODE10](#)
 [hardware.h](#), [169](#)
[STATF_B_OAM](#)
 [hardware.h](#), [169](#)
[STATF_B_VBL](#)
 [hardware.h](#), [169](#)
[STATF_BUSY](#)
 [hardware.h](#), [169](#)
[STATF_HBL](#)
 [hardware.h](#), [168](#)

STATF_INT_VBL
 hardware.h, 183, 196
 STATF_LCD
 hardware.h, 169
 STATF_LYC
 hardware.h, 168
 STATF_LYCF
 hardware.h, 168
 STATF_MODE00
 hardware.h, 168
 STATF_MODE01
 hardware.h, 168
 STATF_MODE10
 hardware.h, 168
 STATF_OAM
 hardware.h, 169
 STATF_SPR_COLL
 hardware.h, 183, 196
 STATF_VBL
 hardware.h, 168
 stdarg.h
 va_arg, 73–75
 va_end, 73–75
 va_list, 74, 75
 va_start, 73, 74
 stdatomic.h
 atomic_flag_clear, 310
 atomic_flag_test_and_set, 310
 stdbool.h
 __bool_true_false_are_defined, 311
 bool, 311
 false, 311
 true, 311
 stddef.h
 __PTRDIFF_T_DEFINED, 311
 __SIZE_T_DEFINED, 311
 __WCHAR_T_DEFINED, 311
 NULL, 311
 offsetof, 311
 ptrdiff_t, 312
 size_t, 312
 wchar_t, 312
 stdint.h
 INT16_C, 315
 INT16_MAX, 313
 INT16_MIN, 313
 int16_t, 316
 INT32_C, 316
 INT32_MAX, 314
 INT32_MIN, 313
 int32_t, 316
 INT8_C, 315
 INT8_MAX, 313
 INT8_MIN, 313
 int8_t, 316
 INT_FAST16_MAX, 314
 INT_FAST16_MIN, 314
 int_fast16_t, 317
 INT_FAST32_MAX, 315
 INT_FAST32_MIN, 314
 int_fast32_t, 317
 INT_FAST8_MAX, 314
 INT_FAST8_MIN, 314
 int_fast8_t, 317
 INT_LEAST16_MAX, 314
 INT_LEAST16_MIN, 314
 int_least16_t, 317
 INT_LEAST32_MAX, 314
 INT_LEAST32_MIN, 314
 int_least32_t, 317
 INT_LEAST8_MAX, 314
 INT_LEAST8_MIN, 314
 int_least8_t, 317
 INTMAX_C, 316
 INTMAX_MAX, 315
 INTMAX_MIN, 315
 intmax_t, 317
 INTPTR_MAX, 315
 INTPTR_MIN, 315
 intptr_t, 317
 PTRDIFF_MAX, 315
 PTRDIFF_MIN, 315
 SIG_ATOMIC_MAX, 315
 SIG_ATOMIC_MIN, 315
 SIZE_MAX, 315
 UINT16_C, 316
 UINT16_MAX, 314
 uint16_t, 316
 UINT32_C, 316
 UINT32_MAX, 314
 uint32_t, 317
 UINT8_C, 316
 UINT8_MAX, 314
 uint8_t, 316
 UINT_FAST16_MAX, 315
 uint_fast16_t, 317
 UINT_FAST32_MAX, 315
 uint_fast32_t, 317
 UINT_FAST8_MAX, 315
 uint_fast8_t, 317
 UINT_LEAST16_MAX, 314
 uint_least16_t, 317
 UINT_LEAST32_MAX, 314
 uint_least32_t, 317
 UINT_LEAST8_MAX, 314
 uint_least8_t, 317
 UINTMAX_C, 316
 UINTMAX_MAX, 315
 uintmax_t, 317
 UINTPTR_MAX, 315
 uintptr_t, 317
 WCHAR_MAX, 316
 WCHAR_MIN, 316
 WINT_MAX, 316
 WINT_MIN, 316
 stdio.h

- getchar, 319
- gets, 319
- printf, 318
- putchar, 318
- puts, 319
- sprintf, 319
- stdlib.h
 - __reentrant, 320
 - abs, 320
 - atoi, 320
 - atol, 321
 - bsearch, 322
 - calloc, 322
 - exit, 320
 - free, 322
 - itoa, 321
 - labs, 320
 - ltoa, 321
 - malloc, 322
 - qsort, 323
 - realloc, 322
 - uitoa, 321
 - ultoa, 322
- stdnoreturn.h
 - noreturn, 323
- strcat
 - string.h, 77, 81, 84
- strcmp
 - string.h, 76, 79, 83
- strcpy
 - string.h, 75, 79, 83
- string.h
 - c, 82
 - memcmp, 78, 82, 86
 - memcpy, 76, 80, 83
 - memmove, 76, 80, 84
 - memset, 76, 80, 84
 - reverse, 77, 80, 84
 - strcat, 77, 81, 84
 - strcmp, 76, 79, 83
 - strcpy, 75, 79, 83
 - strlen, 77, 81, 85
 - strncat, 77, 81, 85
 - strncmp, 78, 81, 85
 - strncpy, 78, 82, 85
- strlen
 - string.h, 77, 81, 85
- strncat
 - string.h, 77, 81, 85
- strncmp
 - string.h, 78, 81, 85
- strncpy
 - string.h, 78, 82, 85
- SVBK_REG
 - hardware.h, 179
- SWITCH_16_8_MODE_MBC1
 - gb.h, 121
- SWITCH_4_32_MODE_MBC1
 - gb.h, 121
- switch_data
 - drawing.h, 105
- SWITCH_RAM
 - gb.h, 121
 - msx.h, 242
 - sms.h, 296
- SWITCH_RAM_MBC1
 - gb.h, 120
- SWITCH_RAM_MBC5
 - gb.h, 122
- SWITCH_ROM
 - gb.h, 120
 - msx.h, 246
 - nes.h, 264
 - sms.h, 296
- SWITCH_ROM1
 - msx.h, 241
 - sms.h, 296
- SWITCH_ROM2
 - msx.h, 241
 - sms.h, 296
- SWITCH_ROM_DUMMY
 - nes.h, 264
- SWITCH_ROM_MBC1
 - gb.h, 120
- SWITCH_ROM_MBC5
 - gb.h, 121
- SWITCH_ROM_MBC5_8M
 - gb.h, 121
- SWITCH_ROM_MEGADUCK
 - gb.h, 120
- sys_time
 - gb.h, 152
 - msx.h, 255
 - nes.h, 284
 - sms.h, 309
- SYSTEM_NTSC
 - hardware.h, 187, 200
- SYSTEM_PAL
 - hardware.h, 187, 200
- TAC_REG
 - hardware.h, 176
- TACF_16KHZ
 - hardware.h, 164
- TACF_262KHZ
 - hardware.h, 164
- TACF_4KHZ
 - hardware.h, 164
- TACF_65KHZ
 - hardware.h, 164
- TACF_START
 - hardware.h, 163
- TACF_STOP
 - hardware.h, 163
- tile
 - OAM_item_t, 71
- TIM_IFLAG

- gb.h, 116
- msx.h, 238
- sms.h, 293
- TIMA_REG
 - hardware.h, 176
- time
 - time.h, 324
- time.h
 - clock, 324
 - CLOCKS_PER_SEC, 324
 - time, 324
 - time_t, 324
- time_t
 - time.h, 324
- TMA_REG
 - hardware.h, 176
- TO_FAR_PTR
 - far_ptr.h, 224
- to_far_ptr
 - far_ptr.h, 226
- tolower
 - ctype.h, 94
- toupper
 - ctype.h, 94
- TRUE
 - types.h, 92
- true
 - stdbool.h, 311
- typeof.h
 - TYPEOF_ARRAY, 325
 - TYPEOF_BIT, 325
 - TYPEOF_BITFIELD, 325
 - TYPEOF_CHAR, 325
 - TYPEOF_CPOINTER, 326
 - TYPEOF_EEPPPOINTER, 326
 - TYPEOF_FIXED16X16, 325
 - TYPEOF_FLOAT, 325
 - TYPEOF_FPOINTER, 326
 - TYPEOF_FUNCTION, 325
 - TYPEOF_GPOINTER, 326
 - TYPEOF_INT, 325
 - TYPEOF_IPOINTER, 326
 - TYPEOF_LONG, 325
 - TYPEOF_POINTER, 326
 - TYPEOF_PPOINTER, 326
 - TYPEOF_SBIT, 325
 - TYPEOF_SFR, 325
 - TYPEOF_SHORT, 325
 - TYPEOF_STRUCT, 325
 - TYPEOF_VOID, 325
- TYPEOF_ARRAY
 - typeof.h, 325
- TYPEOF_BIT
 - typeof.h, 325
- TYPEOF_BITFIELD
 - typeof.h, 325
- TYPEOF_CHAR
 - typeof.h, 325
- TYPEOF_CPOINTER
 - typeof.h, 326
- TYPEOF_EEPPPOINTER
 - typeof.h, 326
- TYPEOF_FIXED16X16
 - typeof.h, 325
- TYPEOF_FLOAT
 - typeof.h, 325
- TYPEOF_FPOINTER
 - typeof.h, 326
- TYPEOF_FUNCTION
 - typeof.h, 325
- TYPEOF_GPOINTER
 - typeof.h, 326
- TYPEOF_INT
 - typeof.h, 325
- TYPEOF_IPOINTER
 - typeof.h, 326
- TYPEOF_LONG
 - typeof.h, 325
- TYPEOF_POINTER
 - typeof.h, 326
- TYPEOF_PPOINTER
 - typeof.h, 326
- TYPEOF_SBIT
 - typeof.h, 325
- TYPEOF_SFR
 - typeof.h, 325
- TYPEOF_SHORT
 - typeof.h, 325
- TYPEOF_STRUCT
 - typeof.h, 325
- TYPEOF_VOID
 - typeof.h, 325

- types.h
- __SIZE_T_DEFINED, 87, 88, 91
- AT, 89
- BANKED, 89
- BOOLEAN, 89
- BYTE, 89
- clock_t, 87, 88, 91
- CRITICAL, 89
- DWORD, 90
- FALSE, 92
- fixed, 90
- INT16, 87, 88, 91
- INT32, 87, 88, 91
- INT8, 87, 88, 91
- INTERRUPT, 89
- LWORD, 90
- NAKED, 89
- NONBANKED, 89
- NULL, 92
- OLDCALL, 89
- POINTER, 92
- PRESERVES_REGS, 89
- SFR, 89
- size_t, 87, 88, 91

- TRUE, [92](#)
- UBYTE, [90](#)
- UDWORD, [90](#)
- UINT16, [87](#), [88](#), [91](#)
- UINT32, [87](#), [88](#), [91](#)
- UINT8, [87](#), [88](#), [91](#)
- ULWORD, [90](#)
- UWORD, [90](#)
- WORD, [90](#)
- Z88DK_CALLEE, [90](#)
- Z88DK_FASTCALL, [91](#)
- UBYTE
 - types.h, [90](#)
- UCHAR_MAX
 - limits.h, [232](#)
- UDWORD
 - types.h, [90](#)
- UINT16
 - types.h, [87](#), [88](#), [91](#)
- UINT16_C
 - stdint.h, [316](#)
- UINT16_MAX
 - stdint.h, [314](#)
- uint16_t
 - stdint.h, [316](#)
- uint2bcd
 - bcd.h, [95](#)
- UINT32
 - types.h, [87](#), [88](#), [91](#)
- UINT32_C
 - stdint.h, [316](#)
- UINT32_MAX
 - stdint.h, [314](#)
- uint32_t
 - stdint.h, [317](#)
- UINT8
 - types.h, [87](#), [88](#), [91](#)
- UINT8_C
 - stdint.h, [316](#)
- UINT8_MAX
 - stdint.h, [314](#)
- uint8_t
 - stdint.h, [316](#)
- UINT_FAST16_MAX
 - stdint.h, [315](#)
- uint_fast16_t
 - stdint.h, [317](#)
- UINT_FAST32_MAX
 - stdint.h, [315](#)
- uint_fast32_t
 - stdint.h, [317](#)
- UINT_FAST8_MAX
 - stdint.h, [315](#)
- uint_fast8_t
 - stdint.h, [317](#)
- UINT_LEAST16_MAX
 - stdint.h, [314](#)
- uint_least16_t
 - stdint.h, [317](#)
- UINT_LEAST32_MAX
 - stdint.h, [314](#)
- uint_least32_t
 - stdint.h, [317](#)
- UINT_LEAST8_MAX
 - stdint.h, [314](#)
- uint_least8_t
 - stdint.h, [317](#)
- UINT_MAX
 - limits.h, [233](#)
- UINT_MIN
 - limits.h, [233](#)
- UINTMAX_C
 - stdint.h, [316](#)
- UINTMAX_MAX
 - stdint.h, [315](#)
- uintmax_t
 - stdint.h, [317](#)
- UINTPTR_MAX
 - stdint.h, [315](#)
- uintptr_t
 - stdint.h, [317](#)
- uitoa
 - stdlib.h, [321](#)
- ULONG_MAX
 - limits.h, [233](#)
- ULONG_MIN
 - limits.h, [233](#)
- ultoa
 - stdlib.h, [322](#)
- ULWORD
 - types.h, [90](#)
- UNSIGNED
 - drawing.h, [104](#)
- USE_C_MEMCPY
 - provides.h, [72](#), [73](#)
- USE_C_STRCMP
 - provides.h, [72](#), [73](#)
- USE_C_STRCPY
 - provides.h, [72](#), [73](#)
- USHRT_MAX
 - limits.h, [233](#)
- USHRT_MIN
 - limits.h, [233](#)
- UWORD
 - types.h, [90](#)
- va_arg
 - stdarg.h, [73–75](#)
- va_end
 - stdarg.h, [73–75](#)
- va_list
 - stdarg.h, [74](#), [75](#)
- va_start
 - stdarg.h, [73](#), [74](#)
- VBK_ATTRIBUTES
 - hardware.h, [170](#), [187](#), [200](#)
- VBK_BANK_0

- hardware.h, [170](#)
- VBK_BANK_1
 - hardware.h, [170](#)
- VBK_REG
 - hardware.h, [178](#)
 - msx.h, [237](#)
 - sms.h, [291](#)
- VBK_TILES
 - hardware.h, [170](#), [187](#), [200](#)
- VBL_IFLAG
 - gb.h, [115](#)
 - msx.h, [238](#)
 - sms.h, [292](#)
- VDP_ATTR_SHIFT
 - hardware.h, [189](#), [202](#)
- VDP_R0
 - hardware.h, [183](#), [196](#)
- VDP_R1
 - hardware.h, [184](#), [197](#)
- VDP_R10
 - hardware.h, [186](#), [199](#)
- VDP_R2
 - hardware.h, [185](#), [197](#)
- VDP_R3
 - hardware.h, [185](#), [198](#)
- VDP_R4
 - hardware.h, [185](#), [198](#)
- VDP_R5
 - hardware.h, [185](#), [198](#)
- VDP_R6
 - hardware.h, [185](#), [198](#)
- VDP_R7
 - hardware.h, [186](#), [198](#)
- VDP_R8
 - hardware.h, [186](#), [199](#)
- VDP_R9
 - hardware.h, [186](#), [199](#)
- VDP_RBORDER
 - hardware.h, [186](#), [198](#)
- VDP_REG_MASK
 - hardware.h, [183](#), [196](#)
- VDP_RSCX
 - hardware.h, [186](#), [199](#)
- VDP_RSCY
 - hardware.h, [186](#), [199](#)
- VDP_SAT_TERM
 - hardware.h, [187](#), [200](#)
- VECTOR_JOYPAD
 - isr.h, [202](#)
- VECTOR_SERIAL
 - isr.h, [202](#)
- VECTOR_STAT
 - isr.h, [202](#)
- VECTOR_TIMER
 - isr.h, [202](#)
- version.h
 - __GBDK_VERSION, [232](#)
- vmemcpy
 - gb.h, [148](#)
 - msx.h, [249](#)
 - sms.h, [303](#)
- vmemset
 - gb.h, [151](#)
 - nes.h, [283](#)
- w
 - _fixed, [66](#)
- wait_int_handler
 - gb.h, [127](#)
- wait_vbl_done
 - gb.h, [130](#)
 - msx.h, [245](#)
 - nes.h, [270](#)
 - sms.h, [300](#)
- waitpad
 - gb.h, [128](#)
 - msx.h, [246](#)
 - nes.h, [268](#)
 - sms.h, [300](#)
- waitpadup
 - gb.h, [129](#)
 - msx.h, [246](#)
 - nes.h, [268](#)
 - sms.h, [300](#)
- WCHAR_MAX
 - stdint.h, [316](#)
- WCHAR_MIN
 - stdint.h, [316](#)
- wchar_t
 - stddef.h, [312](#)
- WHITE
 - drawing.h, [104](#)
- WINT_MAX
 - stdint.h, [316](#)
- WINT_MIN
 - stdint.h, [316](#)
- WORD
 - types.h, [90](#)
- WRITE_VDP_CMD
 - msx.h, [243](#)
 - sms.h, [298](#)
- WRITE_VDP_DATA
 - msx.h, [243](#)
 - sms.h, [298](#)
- wrtchr
 - drawing.h, [106](#)
- WX_REG
 - hardware.h, [178](#)
- WY_REG
 - hardware.h, [178](#)
- x
 - OAM_item_t, [71](#)
- XOR
 - drawing.h, [103](#)
- y

OAM_item_t, [71](#)

Z88DK_CALLEE

types.h, [90](#)

Z88DK_FASTCALL

types.h, [91](#)