

# GBDK 2020 Docs

4.3.0

Generated by Doxygen 1.9.1

Sat Jun 8 2024 00:17:14

<b>1 General Documentation</b>	<b>1</b>
1.1 Introduction	2
1.2 About the Documentation	2
1.3 About GBDK	2
1.4 Historical Info and Links	2
<b>2 Getting Started</b>	<b>2</b>
2.1 1. Download a Release and unzip it	3
2.1.1 Known Issue: Windows and folder names with spaces on non-C drives	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	4
2.4 4. If you use GBTD / GBMB, get the fixed version	4
2.5 5. Review Coding Guidelines	4
2.6 6. Hardware and Resources	4
2.7 7. Set up C Source debugging	4
2.8 8. Try a GBDK Tutorial	5
2.9 9. Read up!	5
2.10 10. Need help?	5
2.11 Migrating From Pre-GBDK-2020 Tutorials	5
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)	6
2.11.7 GBDK Header include changes	6
2.11.8 Include .h headers, not .c source files	6
2.11.9 Use the Template Projects	6
2.11.10 Use hUGTracker instead of gbt_player	6
<b>3 Links, Tools and Debugging</b>	<b>6</b>
3.1 SDCC Compiler Suite User Manual	6
3.2 Getting Help	6
3.3 Game Boy Documentation	7
3.4 Sega Master System / Game Gear Documentation	7
3.5 Tutorials	7
3.6 Example code	7
3.7 Graphics Tools	7
3.8 Music And Sound Effects	8
3.9 Emulators	8
3.10 Debugging tools	8

3.11 Optimizing Assembly . . . . .	9
3.12 Continuous Integration and Deployment . . . . .	9
<b>4 Using GBDK</b>	<b>9</b>
4.1 Interrupts . . . . .	9
4.1.1 Available Interrupts . . . . .	10
4.1.2 Adding your own interrupt handler . . . . .	10
4.1.3 Using your own Interrupt Dispatcher . . . . .	10
4.1.4 Returning from Interrupts and STAT mode . . . . .	11
4.2 What GBDK does automatically and behind the scenes . . . . .	11
4.2.1 NES console . . . . .	11
4.2.2 OAM (VRAM Sprite Attribute Table) . . . . .	11
4.2.3 Graphics Tile Maps and Data on Startup . . . . .	11
4.2.4 Font tiles when using stdio.h . . . . .	11
4.2.5 Default Interrupt Service Handlers (ISRs) . . . . .	11
4.2.6 Ensuring Safe Access to Graphics Memory . . . . .	11
4.3 Compression . . . . .	12
4.4 Copying Functions to RAM and HIRAM . . . . .	12
4.5 Mixing C and Assembly . . . . .	12
4.5.1 Inline ASM within C source files . . . . .	12
4.5.2 In Separate ASM files . . . . .	13
4.6 Including binary files in C source with incbin . . . . .	13
4.7 Known Issues and Limitations . . . . .	13
4.7.1 SDCC . . . . .	13
<b>5 Coding Guidelines</b>	<b>14</b>
5.1 Learning C / C fundamentals . . . . .	14
5.1.1 General C tutorials . . . . .	14
5.1.2 Embedded C introductions . . . . .	14
5.1.3 Game Boy games in C . . . . .	14
5.2 Understanding the hardware . . . . .	14
5.3 Writing optimal C code for the Game Boy and SDCC . . . . .	14
5.3.1 Tools . . . . .	14
5.3.2 Avoid Reading from VRAM . . . . .	15
5.3.3 Variables . . . . .	15
5.3.4 Code structure . . . . .	16
5.3.5 GBDK API/Library . . . . .	16
5.3.6 Toolchain . . . . .	17
5.3.7 Constants, Signed-ness and Overflows . . . . .	17
5.3.8 Chars and vararg functions . . . . .	18
5.4 When C isn't fast enough . . . . .	18
5.4.1 Reusable Local Labels and Inline ASM . . . . .	18
5.4.2 Variables and registers . . . . .	18

5.4.3 Segments / Areas . . . . .	19
5.4.4 Calling convention . . . . .	19
<b>6 ROM/RAM Banking and MBCs</b>	<b>20</b>
6.1 ROM/RAM Banking and MBCs (Memory Bank Controllers) . . . . .	20
6.1.1 Non-banked cartridges . . . . .	21
6.1.2 MBC Banked cartridges (Memory Bank Controllers) . . . . .	21
6.1.3 Recommended MBC type . . . . .	21
6.2 Working with Banks . . . . .	21
6.2.1 Setting the ROM bank for a Source file . . . . .	21
6.2.2 Setting the RAM bank for a Source file . . . . .	22
6.2.3 Setting the MBC and number of ROM & RAM banks available . . . . .	22
6.2.4 MBC Type Chart . . . . .	22
6.2.5 Getting Bank Numbers . . . . .	23
6.2.6 Banking and Functions . . . . .	23
6.2.7 Const Data (Variables in ROM) . . . . .	24
6.2.8 Variables in RAM . . . . .	24
6.2.9 Far Pointers . . . . .	25
6.2.10 Bank switching . . . . .	25
6.2.11 Wrapper Function for Accessing Banked Data . . . . .	25
6.2.12 Currently active bank: CURRENT_BANK . . . . .	25
6.3 Auto-Banking . . . . .	25
6.4 Errors related to banking (overflow, multiple writes to same location) . . . . .	26
6.5 Bank space usage . . . . .	26
6.5.1 Other important notes . . . . .	26
6.6 Banking example projects . . . . .	27
6.7 SMS/Game Gear Banking . . . . .	27
6.7.1 Auto-Banking . . . . .	27
<b>7 GBDK Toolchain</b>	<b>28</b>
7.1 Overview . . . . .	28
7.2 Data Types . . . . .	28
7.3 Changing Important Addresses . . . . .	28
7.4 Compiling programs . . . . .	29
7.4.1 Makefiles . . . . .	29
7.4.2 Using Makefiles . . . . .	29
7.4.3 Linker Files and ROM Auto Banking . . . . .	30
7.5 Build Tools . . . . .	30
7.5.1 lcc . . . . .	30
7.5.2 sdcc . . . . .	30
7.5.3 sdasgb . . . . .	30
7.5.4 bankpack . . . . .	31
7.5.5 sldlgb . . . . .	31

7.5.6 ihxcheck . . . . .	31
7.5.7 makebin . . . . .	31
7.6 GBDK Utilities . . . . .	31
7.6.1 GBCompress . . . . .	31
7.6.2 png2asset . . . . .	32
7.6.3 makecom . . . . .	33
7.6.4 png2hicolorgb . . . . .	33
7.6.5 romusage . . . . .	34
<b>8 Supported Consoles &amp; Cross Compiling</b>	<b>34</b>
8.1 Consoles Supported by GBDK . . . . .	34
8.2 Cross Compiling for Different Consoles . . . . .	34
8.2.1 lcc . . . . .	34
8.2.2 sdcc . . . . .	35
8.2.3 Console Port and Platform Settings . . . . .	35
8.3 Cross-Platform Constants . . . . .	36
8.3.1 Console Identifiers . . . . .	36
8.3.2 Console Hardware Properties . . . . .	36
8.4 Using <gbdk/...> headers . . . . .	37
8.5 Cross Platform Example Projects . . . . .	37
8.5.1 Cross Platform Asset Example . . . . .	37
8.6 Hardware Summaries . . . . .	37
8.6.1 Safe VRAM / Display Controller Access . . . . .	39
8.7 Using Game Boy Color (GBC/CGB) Features . . . . .	39
8.7.1 Differences Versus the Regular Game Boy (DMG/GBP/SGB) . . . . .	39
8.7.2 Game Boy Color features in GBDK . . . . .	39
8.7.3 CGB Examples . . . . .	40
8.8 Porting Between Supported Consoles . . . . .	40
8.8.1 From Game Boy to Analogue Pocket . . . . .	40
8.8.2 From Game Boy to SMS/GG . . . . .	41
8.8.3 From Game Boy to NES . . . . .	42
8.8.4 From Game Boy to Mega Duck / Cougar Boy . . . . .	46
<b>9 Example Programs</b>	<b>48</b>
9.1 banks (various projects) . . . . .	48
9.2 comm . . . . .	48
9.3 crash . . . . .	48
9.4 colorbar . . . . .	48
9.5 dscan . . . . .	48
9.6 filltest . . . . .	49
9.7 fonts . . . . .	49
9.8 galaxy . . . . .	49
9.9 gb-dtmf . . . . .	49

9.10 gbdecompress	49
9.11 irq	49
9.12 large map	49
9.13 metasprites	49
9.14 lcd isr wobble	49
9.15 paint	49
9.16 rand	50
9.17 ram_fn	50
9.18 rpn	50
9.19 samptest	50
9.20 sgb (various)	50
9.21 sound	50
9.22 space	50
9.23 templates	50
<b>10 Frequently Asked Questions (FAQ)</b>	<b>50</b>
10.1 General	50
10.2 Licensing	51
10.3 Graphics and Resources	51
10.4 ROM Header Settings	51
10.5 Editors	51
10.6 Errors and Warnings	52
10.7 Debugging / Compiling / Toolchain	53
10.8 API / Utilities	54
<b>11 Migrating to new GBDK Versions</b>	<b>55</b>
11.1 GBDK-2020 versions	55
11.1.1 Porting to GBDK-2020 4.3.0	55
11.1.2 Porting to GBDK-2020 4.2.0	55
11.1.3 Porting to GBDK-2020 4.1.1	55
11.1.4 Porting to GBDK-2020 4.1.0	56
11.1.5 Porting to GBDK-2020 4.0.6	56
11.1.6 Porting to GBDK-2020 4.0.5	56
11.1.7 Porting to GBDK-2020 4.0.4	57
11.1.8 Porting to GBDK-2020 4.0.3	57
11.1.9 Porting to GBDK-2020 4.0.2	57
11.1.10 Porting to GBDK-2020 4.0.1	57
11.1.11 Porting to GBDK-2020 4.0	57
11.1.12 Porting to GBDK-2020 3.2	57
11.1.13 Porting to GBDK-2020 3.1.1	57
11.1.14 Porting to GBDK-2020 3.1	57
11.1.15 Porting to GBDK-2020 3.0.1	58
11.2 Historical GBDK versions	58

11.2.1 GBDK 1.1 to GBDK 2.0 . . . . .	58
<b>12 GBDK Release Notes</b>	<b>58</b>
12.1 GBDK-2020 Release Notes . . . . .	58
12.1.1 GBDK-2020 4.3.0 . . . . .	58
12.1.2 GBDK-2020 4.2.0 . . . . .	61
12.1.3 GBDK-2020 4.1.1 . . . . .	63
12.1.4 GBDK-2020 4.1.0 . . . . .	63
12.1.5 GBDK-2020 4.0.6 . . . . .	65
12.1.6 GBDK-2020 4.0.5 . . . . .	66
12.1.7 GBDK-2020 4.0.4 . . . . .	67
12.1.8 GBDK-2020 4.0.3 . . . . .	68
12.1.9 GBDK-2020 4.0.2 . . . . .	69
12.1.10 GBDK-2020 4.0.1 . . . . .	69
12.1.11 GBDK-2020 4.0 . . . . .	70
12.1.12 GBDK-2020 3.2 . . . . .	71
12.1.13 GBDK-2020 3.1.1 . . . . .	71
12.1.14 GBDK-2020 3.1 . . . . .	71
12.1.15 GBDK-2020 3.0.1 . . . . .	71
12.1.16 GBDK-2020 3.0 . . . . .	71
12.2 Historical GBDK Release Notes . . . . .	72
12.2.1 GBDK 2.96 . . . . .	72
12.2.2 GBDK 2.95-3 . . . . .	72
12.2.3 GBDK 2.95-2 . . . . .	72
12.2.4 GBDK 2.95 . . . . .	72
12.2.5 GBDK 2.94 . . . . .	73
12.2.6 GBDK 2.93 . . . . .	73
12.2.7 GBDK 2.92-2 for win32 . . . . .	74
12.2.8 GBDK 2.92 . . . . .	74
12.2.9 GBDK 2.91 . . . . .	74
12.2.10 GBDK 2.1.5 . . . . .	75
12.2.11 GBDK 2.0b11 (DOS binary only) - 24 November 1997 . . . . .	75
12.2.12 GBDK 2.0b10 (DOS binary only) - 6 November 1997 . . . . .	75
12.2.13 GBDK 2.0b9 (DOS binary only) . . . . .	75
12.2.14 GBDK 2.0b8 (DOS binary only) . . . . .	75
12.2.15 GBDK 2.0b7 (DOS binary only) . . . . .	75
12.2.16 GBDK 2.0b6 . . . . .	76
12.2.17 GBDK 2.0b5 . . . . .	76
12.2.18 GBDK 2.0b4 . . . . .	76
12.2.19 GBDK 2.0b3 . . . . .	76
12.2.20 GBDK 2.0b2 . . . . .	76
12.2.21 GBDK 2.0b1 . . . . .	77

12.2.22 GBDK 1.1 . . . . .	77
12.2.23 GBDK 1.0-1 1996 . . . . .	77
<b>13 Toolchain settings</b>	<b>77</b>
13.1 lcc settings . . . . .	77
13.2 sdcc settings . . . . .	78
13.3 sdasgb settings . . . . .	79
13.4 sdasz80 settings . . . . .	80
13.5 sdas6500 settings . . . . .	80
13.6 bankpack settings . . . . .	81
13.7 sldlgb settings . . . . .	81
13.8 slddz80 settings . . . . .	81
13.9 sldd6808 settings . . . . .	82
13.10 ihxcheck settings . . . . .	82
13.11 makebin settings . . . . .	82
13.12 makecom settings . . . . .	83
13.13 gbcompress settings . . . . .	83
13.14 png2asset settings . . . . .	83
13.15 png2hicolorgb settings . . . . .	84
13.16 romusage settings . . . . .	84
<b>14 Todo List</b>	<b>85</b>
<b>15 Module Index</b>	<b>85</b>
15.1 C modules . . . . .	85
<b>16 Data Structure Index</b>	<b>85</b>
16.1 Data Structures . . . . .	85
<b>17 File Index</b>	<b>85</b>
17.1 File List . . . . .	85
<b>18 Module Documentation</b>	<b>88</b>
18.1 List of gbdk fonts . . . . .	88
18.1.1 Description . . . . .	88
18.1.2 Variable Documentation . . . . .	88
<b>19 Data Structure Documentation</b>	<b>88</b>
19.1 __far_ptr Union Reference . . . . .	88
19.1.1 Detailed Description . . . . .	89
19.1.2 Field Documentation . . . . .	89
19.2 _fixed Union Reference . . . . .	89
19.2.1 Detailed Description . . . . .	89
19.2.2 Field Documentation . . . . .	89
19.3 atomic_flag Struct Reference . . . . .	90



19.3.1 Field Documentation . . . . .	90
19.4 isr_nested_vector_t Struct Reference . . . . .	90
19.4.1 Field Documentation . . . . .	90
19.5 isr_vector_t Struct Reference . . . . .	90
19.5.1 Field Documentation . . . . .	91
19.6 joypads_t Struct Reference . . . . .	91
19.6.1 Detailed Description . . . . .	92
19.6.2 Field Documentation . . . . .	92
19.7 metasprite_t Struct Reference . . . . .	92
19.7.1 Detailed Description . . . . .	93
19.7.2 Field Documentation . . . . .	93
19.8 OAM_item_t Struct Reference . . . . .	93
19.8.1 Detailed Description . . . . .	94
19.8.2 Field Documentation . . . . .	94
19.9 sfont_handle Struct Reference . . . . .	94
19.9.1 Detailed Description . . . . .	95
19.9.2 Field Documentation . . . . .	95
<b>20 File Documentation . . . . .</b>	<b>95</b>
20.1 docs/pages/01_getting_started.md File Reference . . . . .	95
20.2 docs/pages/02_links_and_tools.md File Reference . . . . .	95
20.3 docs/pages/03_using_gbdk.md File Reference . . . . .	95
20.4 docs/pages/04_coding_guidelines.md File Reference . . . . .	95
20.5 docs/pages/05_banking_mbc5.md File Reference . . . . .	95
20.6 docs/pages/06_toolchain.md File Reference . . . . .	95
20.7 docs/pages/06b_supported_consoles.md File Reference . . . . .	95
20.8 docs/pages/07_sample_programs.md File Reference . . . . .	95
20.9 docs/pages/08_faq.md File Reference . . . . .	95
20.10 docs/pages/09_migrating_new_versions.md File Reference . . . . .	95
20.11 docs/pages/10_release_notes.md File Reference . . . . .	95
20.12 docs/pages/20_toolchain_settings.md File Reference . . . . .	95
20.13 docs/pages/docs_index.md File Reference . . . . .	95
20.14 gbdk-lib/include/asm/mos6502/provides.h File Reference . . . . .	95
20.14.1 Macro Definition Documentation . . . . .	95
20.15 gbdk-lib/include/asm/sm83/provides.h File Reference . . . . .	96
20.15.1 Macro Definition Documentation . . . . .	96
20.16 gbdk-lib/include/asm/z80/provides.h File Reference . . . . .	96
20.16.1 Macro Definition Documentation . . . . .	96
20.17 gbdk-lib/include/asm/mos6502/stdarg.h File Reference . . . . .	96
20.17.1 Macro Definition Documentation . . . . .	96
20.17.2 Typedef Documentation . . . . .	97
20.18 gbdk-lib/include/asm/sm83/stdarg.h File Reference . . . . .	97

20.18.1 Macro Definition Documentation . . . . .	97
20.18.2 Typedef Documentation . . . . .	97
20.19 gbdk-lib/include/asm/z80/stdarg.h File Reference . . . . .	98
20.19.1 Macro Definition Documentation . . . . .	98
20.19.2 Typedef Documentation . . . . .	98
20.20 gbdk-lib/include/stdarg.h File Reference . . . . .	98
20.21 gbdk-lib/include/asm/mos6502/string.h File Reference . . . . .	98
20.21.1 Detailed Description . . . . .	99
20.21.2 Macro Definition Documentation . . . . .	99
20.21.3 Function Documentation . . . . .	99
20.22 gbdk-lib/include/asm/sm83/string.h File Reference . . . . .	102
20.22.1 Detailed Description . . . . .	102
20.22.2 Function Documentation . . . . .	103
20.22.3 Variable Documentation . . . . .	106
20.23 gbdk-lib/include/asm/z80/string.h File Reference . . . . .	106
20.23.1 Detailed Description . . . . .	106
20.23.2 Function Documentation . . . . .	106
20.24 gbdk-lib/include/string.h File Reference . . . . .	110
20.24.1 Detailed Description . . . . .	110
20.25 gbdk-lib/include/asm/mos6502/types.h File Reference . . . . .	110
20.25.1 Detailed Description . . . . .	110
20.25.2 Macro Definition Documentation . . . . .	110
20.25.3 Typedef Documentation . . . . .	110
20.26 gbdk-lib/include/asm/sm83/types.h File Reference . . . . .	111
20.26.1 Detailed Description . . . . .	111
20.26.2 Macro Definition Documentation . . . . .	111
20.26.3 Typedef Documentation . . . . .	111
20.27 gbdk-lib/include/asm/types.h File Reference . . . . .	112
20.27.1 Detailed Description . . . . .	112
20.27.2 Macro Definition Documentation . . . . .	112
20.27.3 Typedef Documentation . . . . .	113
20.28 gbdk-lib/include/asm/z80/types.h File Reference . . . . .	114
20.28.1 Detailed Description . . . . .	114
20.28.2 Macro Definition Documentation . . . . .	114
20.28.3 Typedef Documentation . . . . .	114
20.29 gbdk-lib/include/types.h File Reference . . . . .	115
20.29.1 Detailed Description . . . . .	115
20.29.2 Macro Definition Documentation . . . . .	115
20.29.3 Typedef Documentation . . . . .	115
20.30 gbdk-lib/include/assert.h File Reference . . . . .	116
20.30.1 Macro Definition Documentation . . . . .	116
20.30.2 Function Documentation . . . . .	116

20.31 gbdk-lib/include/ctype.h File Reference . . . . .	116
20.31.1 Detailed Description . . . . .	116
20.31.2 Function Documentation . . . . .	116
20.32 gbdk-lib/include/gb/bcd.h File Reference . . . . .	118
20.32.1 Detailed Description . . . . .	118
20.32.2 Macro Definition Documentation . . . . .	118
20.32.3 Typedef Documentation . . . . .	118
20.32.4 Function Documentation . . . . .	118
20.33 gbdk-lib/include/gbdk/bcd.h File Reference . . . . .	120
20.34 gbdk-lib/include/nes/bcd.h File Reference . . . . .	121
20.34.1 Detailed Description . . . . .	121
20.34.2 Macro Definition Documentation . . . . .	121
20.34.3 Typedef Documentation . . . . .	121
20.34.4 Function Documentation . . . . .	121
20.35 gbdk-lib/include/sms/bcd.h File Reference . . . . .	122
20.35.1 Detailed Description . . . . .	123
20.35.2 Macro Definition Documentation . . . . .	123
20.35.3 Typedef Documentation . . . . .	123
20.35.4 Function Documentation . . . . .	123
20.36 gbdk-lib/include/gb/bgb_emu.h File Reference . . . . .	124
20.36.1 Detailed Description . . . . .	124
20.37 gbdk-lib/include/gb/cgb.h File Reference . . . . .	124
20.37.1 Detailed Description . . . . .	125
20.37.2 Macro Definition Documentation . . . . .	126
20.37.3 Typedef Documentation . . . . .	128
20.37.4 Function Documentation . . . . .	128
20.38 gbdk-lib/include/gb/crash_handler.h File Reference . . . . .	130
20.38.1 Detailed Description . . . . .	131
20.38.2 Function Documentation . . . . .	131
20.39 gbdk-lib/include/gb/drawing.h File Reference . . . . .	131
20.39.1 Detailed Description . . . . .	132
20.39.2 Macro Definition Documentation . . . . .	132
20.39.3 Function Documentation . . . . .	133
20.40 gbdk-lib/include/gb/emu_debug.h File Reference . . . . .	135
20.40.1 Detailed Description . . . . .	135
20.41 gbdk-lib/include/gbdk/emu_debug.h File Reference . . . . .	136
20.41.1 Detailed Description . . . . .	136
20.41.2 Macro Definition Documentation . . . . .	136
20.41.3 Function Documentation . . . . .	138
20.41.4 Variable Documentation . . . . .	139
20.42 gbdk-lib/include/gb/gb.h File Reference . . . . .	139
20.42.1 Detailed Description . . . . .	143

20.42.2 Macro Definition Documentation . . . . .	143
20.42.3 Typedef Documentation . . . . .	155
20.42.4 Function Documentation . . . . .	155
20.42.5 Variable Documentation . . . . .	189
20.43 gbdk-lib/include/gb/gbdecompress.h File Reference . . . . .	190
20.43.1 Detailed Description . . . . .	190
20.43.2 Function Documentation . . . . .	190
20.44 gbdk-lib/include/gbdk/gbdecompress.h File Reference . . . . .	192
20.45 gbdk-lib/include/sms/gbdecompress.h File Reference . . . . .	192
20.45.1 Function Documentation . . . . .	192
20.45.2 Variable Documentation . . . . .	193
20.46 gbdk-lib/include/gb/hardware.h File Reference . . . . .	193
20.46.1 Detailed Description . . . . .	199
20.46.2 Macro Definition Documentation . . . . .	199
20.46.3 Variable Documentation . . . . .	211
20.47 gbdk-lib/include/msx/hardware.h File Reference . . . . .	216
20.47.1 Detailed Description . . . . .	218
20.47.2 Macro Definition Documentation . . . . .	218
20.47.3 Variable Documentation . . . . .	222
20.48 gbdk-lib/include/nes/hardware.h File Reference . . . . .	223
20.48.1 Detailed Description . . . . .	224
20.48.2 Macro Definition Documentation . . . . .	224
20.48.3 Function Documentation . . . . .	226
20.48.4 Variable Documentation . . . . .	227
20.49 gbdk-lib/include/sms/hardware.h File Reference . . . . .	227
20.49.1 Detailed Description . . . . .	230
20.49.2 Macro Definition Documentation . . . . .	230
20.49.3 Variable Documentation . . . . .	239
20.50 gbdk-lib/include/gb/hblankcpy.h File Reference . . . . .	240
20.50.1 Function Documentation . . . . .	240
20.50.2 Variable Documentation . . . . .	241
20.51 gbdk-lib/include/gb/isr.h File Reference . . . . .	241
20.51.1 Detailed Description . . . . .	242
20.51.2 Macro Definition Documentation . . . . .	242
20.51.3 Typedef Documentation . . . . .	243
20.52 gbdk-lib/include/gb/metasprites.h File Reference . . . . .	243
20.52.1 Detailed Description . . . . .	244
20.52.2 Metasprite support . . . . .	244
20.52.3 Metasprites composed of variable numbers of sprites . . . . .	244
20.52.4 Metasprites and sprite properties (including cgb palette) . . . . .	244
20.52.5 Macro Definition Documentation . . . . .	245
20.52.6 Typedef Documentation . . . . .	245

20.52.7 Function Documentation . . . . .	246
20.52.8 Variable Documentation . . . . .	249
20.53 gbdk-lib/include/gbdk/metasprites.h File Reference . . . . .	250
20.54 gbdk-lib/include/msx/metasprites.h File Reference . . . . .	250
20.54.1 Macro Definition Documentation . . . . .	250
20.54.2 Typedef Documentation . . . . .	251
20.54.3 Function Documentation . . . . .	251
20.54.4 Variable Documentation . . . . .	252
20.55 gbdk-lib/include/nes/metasprites.h File Reference . . . . .	253
20.55.1 Detailed Description . . . . .	253
20.55.2 Metasprite support . . . . .	253
20.55.3 Macro Definition Documentation . . . . .	254
20.55.4 Typedef Documentation . . . . .	254
20.55.5 Function Documentation . . . . .	254
20.55.6 Variable Documentation . . . . .	258
20.56 gbdk-lib/include/sms/metasprites.h File Reference . . . . .	258
20.56.1 Detailed Description . . . . .	259
20.56.2 Metasprite support . . . . .	259
20.56.3 Metasprite support . . . . .	259
20.56.4 Macro Definition Documentation . . . . .	259
20.56.5 Typedef Documentation . . . . .	259
20.56.6 Function Documentation . . . . .	260
20.56.7 Variable Documentation . . . . .	263
20.57 gbdk-lib/include/gb/sgb.h File Reference . . . . .	263
20.57.1 Detailed Description . . . . .	264
20.57.2 Macro Definition Documentation . . . . .	264
20.57.3 Function Documentation . . . . .	265
20.57.4 Variable Documentation . . . . .	266
20.58 gbdk-lib/include/gbdk/console.h File Reference . . . . .	266
20.58.1 Detailed Description . . . . .	266
20.58.2 Function Documentation . . . . .	266
20.59 gbdk-lib/include/gbdk/far_ptr.h File Reference . . . . .	267
20.59.1 Detailed Description . . . . .	268
20.59.2 Macro Definition Documentation . . . . .	268
20.59.3 Typedef Documentation . . . . .	269
20.59.4 Function Documentation . . . . .	269
20.59.5 Variable Documentation . . . . .	270
20.60 gbdk-lib/include/gbdk/font.h File Reference . . . . .	270
20.60.1 Detailed Description . . . . .	270
20.60.2 Macro Definition Documentation . . . . .	271
20.60.3 Typedef Documentation . . . . .	271
20.60.4 Function Documentation . . . . .	271

20.61 gbdk-lib/include/gbdk/gbdk-lib.h File Reference . . . . .	272
20.61.1 Detailed Description . . . . .	272
20.62 gbdk-lib/include/gbdk/incbin.h File Reference . . . . .	272
20.62.1 Detailed Description . . . . .	272
20.62.2 Macro Definition Documentation . . . . .	272
20.63 gbdk-lib/include/gbdk/platform.h File Reference . . . . .	274
20.64 gbdk-lib/include/gbdk/rledecompress.h File Reference . . . . .	274
20.64.1 Detailed Description . . . . .	274
20.64.2 Macro Definition Documentation . . . . .	274
20.64.3 Function Documentation . . . . .	274
20.65 gbdk-lib/include/gbdk/version.h File Reference . . . . .	275
20.65.1 Macro Definition Documentation . . . . .	275
20.66 gbdk-lib/include/limits.h File Reference . . . . .	275
20.66.1 Macro Definition Documentation . . . . .	275
20.67 gbdk-lib/include/msx/msx.h File Reference . . . . .	277
20.67.1 Detailed Description . . . . .	280
20.67.2 Macro Definition Documentation . . . . .	280
20.67.3 Typedef Documentation . . . . .	287
20.67.4 Function Documentation . . . . .	288
20.67.5 Variable Documentation . . . . .	301
20.68 gbdk-lib/include/nes/nes.h File Reference . . . . .	303
20.68.1 Detailed Description . . . . .	306
20.68.2 Macro Definition Documentation . . . . .	306
20.68.3 Typedef Documentation . . . . .	314
20.68.4 Function Documentation . . . . .	314
20.68.5 Variable Documentation . . . . .	340
20.69 gbdk-lib/include/nes/rgb_to_nes_macro.h File Reference . . . . .	341
20.69.1 Macro Definition Documentation . . . . .	341
20.70 gbdk-lib/include/rand.h File Reference . . . . .	341
20.70.1 Detailed Description . . . . .	342
20.70.2 Macro Definition Documentation . . . . .	342
20.70.3 Function Documentation . . . . .	342
20.70.4 Variable Documentation . . . . .	343
20.71 gbdk-lib/include/setjmp.h File Reference . . . . .	343
20.71.1 Macro Definition Documentation . . . . .	343
20.71.2 Typedef Documentation . . . . .	344
20.71.3 Function Documentation . . . . .	344
20.72 gbdk-lib/include/sms/sms.h File Reference . . . . .	344
20.72.1 Detailed Description . . . . .	348
20.72.2 Macro Definition Documentation . . . . .	348
20.72.3 Typedef Documentation . . . . .	356
20.72.4 Function Documentation . . . . .	356

20.72.5 Variable Documentation . . . . .	370
20.73 gbdk-lib/include/stdatomic.h File Reference . . . . .	371
20.73.1 Function Documentation . . . . .	371
20.74 gbdk-lib/include/stdbool.h File Reference . . . . .	372
20.74.1 Macro Definition Documentation . . . . .	372
20.75 gbdk-lib/include/stddef.h File Reference . . . . .	372
20.75.1 Macro Definition Documentation . . . . .	372
20.75.2 Typedef Documentation . . . . .	373
20.76 gbdk-lib/include/stdint.h File Reference . . . . .	373
20.76.1 Macro Definition Documentation . . . . .	374
20.76.2 Typedef Documentation . . . . .	377
20.77 gbdk-lib/include/stdio.h File Reference . . . . .	379
20.77.1 Detailed Description . . . . .	379
20.77.2 Function Documentation . . . . .	379
20.78 gbdk-lib/include/stdlib.h File Reference . . . . .	380
20.78.1 Function Documentation . . . . .	381
20.79 gbdk-lib/include/stdnoreturn.h File Reference . . . . .	384
20.79.1 Macro Definition Documentation . . . . .	384
20.80 gbdk-lib/include/time.h File Reference . . . . .	384
20.80.1 Detailed Description . . . . .	384
20.80.2 Macro Definition Documentation . . . . .	384
20.80.3 Typedef Documentation . . . . .	384
20.80.4 Function Documentation . . . . .	385
20.81 gbdk-lib/include/typeof.h File Reference . . . . .	385
20.81.1 Macro Definition Documentation . . . . .	386
<b>Index</b>	<b>389</b>

## 1 General Documentation

- [Getting Started](#)
- [Links, Tools and Debugging](#)
- [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 and NES

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](http://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>

### 2.1.1 Known Issue: Windows and folder names with spaces on non-C drives

There is a known issue on Windows where sdcc will fail when run from folder names with spaces on non-C drives. For the time being the workaround is as follows (with `D:\My Stuff\` as an example folder):

- Run Windows Command as administrator
- Run: `fsutil.exe 8dot3name query D:`
  - Output: The volume state is: 1 (8dot3 name creation is disabled). The registry state is: 2 (Per volume setting - the default). Based on the above settings, 8dot3 name creation is disabled on D:
- Run: `fsutil 8dot3name set D: 0`
  - Output: Successfully enabled 8dot3name generation on D:
- Run: `fsutil.exe 8dot3name query D:`
  - Output: The volume state is: 0 (8dot3 name creation is enabled). The registry state is: 2 (Per volume setting - the default). Based on the above settings, 8dot3 name creation is enabled on D:
- Only folders created AFTER the setting has been enabled will get 8.3 filename support, renaming folders does NOT appear to generate 8.3 filename support. However it is possible to manually generate a short path name for an existing folder:
  - Run: `D:\>fsutil file setshortname "D:\My stuff" "mystuf~1"`

## 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`.

**2.2.2.1 macOS security warnings** If you get a security warning on macOS that says ("... developer cannot be verified, macOS cannot verify that this app is free from malware"), it does not mean that GBDK is malware. It just means the GBDK toolchain binaries are not signed by Apple, so it won't run them without an additional step. You will need to unquarantine the files in the bin folder in order to run them. This can be fixed using the following steps.

Open a terminal and navigate to the gbdk bin folder ("`bin/`" under your GBDK-2020 install folder). Then type:

```
xattr -d com.apple.quarantine *
```

## 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](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 [hUGEdriver](#) are smaller, more efficient and more versatile than `gbt_player`.

## 3 Links, Tools and Debugging

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 League Tutorial**

<https://blog.ty-porter.dev/development/2021/04/04/writing-a-gameboy-game-in-2021-pt1.html>

### 3.6 Example code

- **Simplified GBDK examples**

[https://github.com/mrombout/gbdk\\_playground/commits/master](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](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

- **hUGTracker 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/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

- **Emulicious**

An accurate emulator with extensive tools including source level debugging. <https://emulicious.net/>

- **BGB**

Accurate emulator, has useful debugging tools.

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

Intellisense in VSCode may have trouble identifying some GBDK types or functions, and therefore flag them as warnings or unidentified.

GBDK platform constants can be declared so that header files are parsed more completely in VSCode. The following `c_cpp_properties.json` example may be adapted for your own project.

```
{
  "configurations": [
    {
      "name": "gameboy",
      "includePath": [
        "${workspaceFolder}/src/**",
        "${workspaceFolder}/res/**",
        "${workspaceFolder}/include/**",
        "${workspaceFolder}/../../../../gbdk/include/**"
      ],
      "defines": ["__PORT_sm83", "__TARGET_gb"],
      "compilerPath": "",
      "cStandard": "c11",
      "intelliSenseMode": "${default}",
      "compilerArgs": [],
      "browse": {
        "limitSymbolsToIncludedHeaders": true
      }
    }
  ],
  "version": 4
}
```

### 3.10 Debugging tools

- **Emulicious debug adapter**

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

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

- If compiler optimization is making the program source hard to step through in the debugger then adding this flag to `lcc` can help. Note that using this flag will likely reduce code performance and increase code size while enabled, so it is best to only use it temporarily.

```
* -Wf--max-allocs-per-node0
```

- **romusage**

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

See [romusage-settings](#)

- **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/wujoyd/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 ([send\\_byte\(\)](#), [receive\\_byte\(\)](#)).
  - Once installed the default SIO ISR cannot be removed. 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 NES console

For implementation details on the NES console in GBDK, see the [NES entry](#) in [Supported Consoles & Cross Compiling](#)

### 4.2.2 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.

### 4.2.3 Graphics Tile Maps and Data on Startup

By default for the Game Boy GBDK assigns:

- Background and Window Tile data starting at `0x8800`
- Background Tile Map starting at `0x9800`
- Window Tile Map starting at `0x9C00`
- Sprites to `8x8` mode

### 4.2.4 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.

### 4.2.5 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.

### 4.2.6 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](https://gbdev.io/pandocs/Accessing_VRAM_and_OAM.html)

### 4.3 Compression

For programs that would benefit from compression GBDK includes the [gbcompress](#) utility and companion API functions.

In addition to the built-in compression `unapack` is another option:

- UnPACK aPack decompression by Toxa: <https://github.com/untoxa>
- apultra aPack compression: <https://github.com/emmanuel-marty/apultra>

Another way to save space is using 1 bit-per-pixel (bpp) tile pattern data instead of 2-bpp or 4-bpp data. This can reduce the ROM size for groups of tiles which only require two shades of color.

- See: [set\\_1bpp\\_colors\(\)](#), [set\\_bkg\\_1bpp\\_data\(\)](#), [set\\_win\\_1bpp\\_data\(\)](#), [set\\_sprite\\_1bpp\\_data\(\)](#)

Use of 1-bpp tile pattern data may be combined with the compression described above to save even more space, however that approach requires using an intermediary RAM buffer before the tile pattern data can be written to VRAM with the `set_*_1bpp_data()` functions.

### 4.4 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.

### 4.5 Mixing C and Assembly

*The following is primarily oriented toward the Game Boy and related clones (sm83 devices), other targets such as sms/gg may vary.*

You can mix C and assembly (ASM) in two ways as described below.

- For additional detail see the [links\\_sdcc\\_docs](#) and [SDCC Calling Conventions](#).

#### 4.5.1 Inline ASM within C source files

- The optional `NAKED` keyword may be used to indicate that the function setup and return should have no handling done by the compiler, and will instead be handled entirely by user code.
- If the entire function preserves some registers the optional `PRESERVES_REGS` keyword may be used as additional hinting for the compiler. For example `PRESERVES_REGS(b, c)`. By default it is assumed by the compiler that no registers are preserved.

Example:

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

Another Example:

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

### 4.5.2 In Separate ASM files

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.
- Parameters will be passed, registers saved and results returned in a manner based on the [SDCC Calling Convention](#) used and how the function is declared.
- Assembly identifiers are exported using the `.globl` directive.
- See `global.s` for examples of hardware register definitions.

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

`main.c`

```
uint16_t add(uint16_t, uint16_t);
```

```
main()
```

```
{
    uint16_t i;

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

`add.s`

```
.globl _add

.area _CODE
_add:      ; uint16_t add(uint16_t First, uint16_t Second)
           ;
           ; In this particular example there is no use and modification of the stack
           ; No need to save and restore registers
           ;
           ; For calling convention __sdcccall(1)
           ; - first 16 bit param is passed in DE
           ; - second 16 bit param is passed in BC

; Load Second Parameter ("Second") into HL
ld  l,  c
ld  h,  b

; Add Parameters "Second" + "First"
add hl, de

; Return result in BC
ld  c,  l
ld  b,  h
ret          ; 16 bit values are returned in BC
```

## 4.6 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.

## 4.7 Known Issues and Limitations

### 4.7.1 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.
- In SDCC **pre-initializing a variable** assigned to SRAM with `-Wf-ba*` will force that variable to be in WRAM instead.

- The following is a workaround for initializing a variable in SRAM. It assigns value 0xA5 to a variable in bank 0 and assigned to address 0xA000 using the `AT()` directive:

```
// Workaround for initializing variable in SRAM
// (MBC RAM and Bank needs to get enabled during GSINIT loading)
static uint8_t AT(0x0000) __rRAMG = 0x0a; // Enable SRAM
static uint8_t AT(0x4000) __rRAMB = 0x00; // Set SRAM bank 0
// Now SRAM is enabled so the variable can get initialized
uint8_t AT(0xA000) initialized_sram_var = 0xA5u;
```

## 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 Avoid Reading from VRAM

In general avoid reading from VRAM since that memory is not accessible at all times. If GBDK a API function which reads from VRAM (such as `get_bkg_tile_xy()`) is called during a video mode when VRAM is not accessible, then that function call will delay until VRAM becomes accessible again. This can cause unnecessary slowdowns when running programs on the Game Boy. It is also not supported by GBDK on the NES platform.

Instead it is better to store things such as map data in general purpose RAM which does not have video mode access limitations.

For more information about video modes and VRAM access see the pan docs:

<https://gbdev.io/pandocs/STAT.html#stat-modes>

### 5.3.3 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).
  - An exception to this when there are a small number of local variables (one or two) and the code is not complex. Then the compiler may allocate those variables to CPU registers instead which may be faster.
  - Functions which use global or static local variables will loose re-entrancy. In most cases it is not a problem, but important to keep in mind.
  - In particular avoid putting big arrays on the stack, consider static local or global.
- Keep the number of arguments passed to functions small (ideally one or two arguments at most). When there are a large number of arguments they get pushed onto the stack and result in more overhead for function calls. See the Calling Conventions in the SDCC compiler manual for details.
- `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-constant>
- For calculated values that don't change, pre-compute results once and store the result. Using lookup-tables and similar approaches 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);
```

### 5.3.4 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](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, `vsync()` 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.

### 5.3.5 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 `vsync()` (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).

### 5.3.6 Toolchain

- See SDCC optimizations: <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.8.1>
- For details about default Compiler data types, see the SDCC Manual (follow links and scroll down 1 page)
  - <https://sdcc.sourceforge.net/doc/sdccman.pdf#section.1.1>
  - Note: by default GBDK enables `--fsigned-char` (via `lcc`) for SDCC
- 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.

### 5.3.7 Constants, Signed-ness and Overflows

There are 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 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.  

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

### 5.3.8 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:

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

Instead this will give the intended output:

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

**5.3.8.1 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

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 Reusable Local Labels and Inline ASM

When functions are written assembly it's generally better to not mix the inline ASM with C code and instead write the whole function in assembly.

If they are mixed then descriptive named labels should not be used for inline ASM. This is due to descriptive labels interfering with the expected scope of the reusable local labels generated from the compiled C code. The compiler will not detect this problem and the resulting code may fail to execute correctly without warning.

Instead use reusable local symbols/labels (for example `1$:`). To learn more about them check the SDAS manual section "1.3.3 Reusable Symbols"

### 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

### 5.4.4 Calling convention

*The following is primarily oriented toward the Game Boy and related clones (sm83 devices), other targets such as sms/gg may vary.*

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 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 additional 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>
- Section 4.3.9 isn't specific about it, but gbz80/sm83 generally share this subheading with z80 (Game Boy is partially a sub-port of z80 in SDCC). <https://sdcc.sourceforge.net/doc/sdccman.pdf#subsection.4.3.9>

**5.4.4.1 Banked Calling Convention** *The following is primarily oriented toward the Game Boy and related clones (sm83 devices), other targets such as sms/gg may vary.*

Key Points:

- Function arguments (if present) are always placed on the stack, right to left without particular alignment
- A fixed stack offset (sm83:+4, z80:+3) is added by the Callee (to skip the pushed Caller Bank and additional Trampoline Return Address)

- Return values follow the calling convention (`__sdcccall(1)`, or `__sdcccall(0)` for `OLDCALL`)

Terminology:

- **Caller:** the code which is calling the requested function
- **Callee:** the function to be called (declared as `BANKED` or `__banked`)
- **Trampoline:** The intermediary which performs the bank switching and does hand-off between **Caller** and **Callee** during the call and then return.

Banked Call Trampoline

- Banked calls are performed via a trampoline in the non-banked region 0000-3ffff
- The `__sdcc_bcall_ehl` trampoline is used by default
  - With it both calling conventions are supported: `__sdcccall(1)` (default) or `__sdcccall(0)` for `OLDCALL`
- If `--legacy-banking` is specified to SDCC the `__sdcc_bcall` trampoline is used.
  - This may only be used with `__sdcccall(0)`

Process for a banked call (using `__sdcc_bcall_ehl`, the default)

#### 1. The Caller

- Function arguments (if present) are always placed on the stack, right to left without particular alignment
- The Bank of Callee function is placed into register E
- The Address of Callee function is placed in HL
- Calls the bank switch Trampoline (which adds Caller return address to the stack)

#### 2. The Trampoline

- Saves the Current Bank onto the stack (pushed as AF, so 16 bits)
- Switches to the Bank of Callee function (in register E)
- Calls the Callee function address in HL (which adds Trampoline return address to the stack)

#### 3. The Callee Function

- SDCC will use an offset to skip the first N bytes of the stack
  - For `sm83` (GB/AP/DUCK): skip first 4 bytes
  - For `z80` (GG/SMS/etc): skip first 3 bytes
- Return values follow the calling convention (`__sdcccall(1)`, or `__sdcccall(0)` for `OLDCALL`)
- Executes a return to Trampoline

#### 4. The Trampoline

- Switches to the Bank of the Caller saved on the stack (and moves Stack Pointer past it)
- Executes a return to Caller

#### 5. The Caller

- Cleans up the stack and uses return value (if present)

## 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).

The majority of this section about banking is focused on the Game Boy since that is the original GBDK platform. Much of it still applies for the Game Gear(GG) and Sega Master System(SMS). For additional details about banking specifically related to these two systems see the [SMS/GG Banking](#) section.

### 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 <= 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 it's 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](#)

For SMS/GG, the ROM file size must be at least 64K to enable mapper support for RAM banks in emulators.

- If the generated ROM is too small then `-yo 4` for makebin (or `-Wm-yo4` for LCC) can be used to set the size to 64K.

### 6.2.4 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

Hex Code	MBC Type	SRAM	Battery	RTC	Rumble	Extra	Max ROM Size (1)
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.

### 6.2.5 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()`.

### 6.2.6 Banking and Functions

#### 6.2.6.1 BANKED/NONBANKED Keywords for Functions

- `BANKED` (is a calling convention):
  - The function will use banked (`far`) sdcc calls (which switch to the function's ROM bank automatically).
  - 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`/ banked sdcc calls)
  - Placed in the bank selected by its source file (or compiler switches).

#### 6.2.6.2 Banked Function Calls

Functions in banks 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 fixed Bank 0 and the currently active bank.
  - A [NONBANKED wrapper function](#) may be used to access data in other banks.
  - Banks cannot be switched manually from inside a `BANKED` function (otherwise it will switch out it's own function code as it is executing it, likely leading to a crash).

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).

Calling Convention:

- For details see [Banked Calling Convention](#)

#### 6.2.7 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.

#### 6.2.8 Variables in RAM

**Todo** Variables in RAM

### 6.2.9 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.

### 6.2.10 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`.

### 6.2.11 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));
```

### 6.2.12 Currently active bank: CURRENT\_BANK

The global variable [CURRENT\\_BANK](#) (a macro for `__current_bank`) is updated automatically when calling [SWITCH\\_ROM\(\)](#), [SWITCH\\_ROM\\_MBC1\(\)](#) and [SWITCH\\_ROM\\_MBC5](#), or when a `BANKED` function is called.

Normally 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);
```

## 6.3 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.

## 6.4 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 [romusage](#) tool.

## 6.5 Bank space usage

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

### 6.5.1 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.



## 6.6 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.

"SMS/GG Banking" section.

## 6.7 SMS/Game Gear Banking

The memory banking setup for SMS and Game Gear in GBDK is different than it is for the Game Boy. Instead of a single switchable bank in the `0x4000 - 0x7FFF` range, there are two switchable frames at different address ranges. The configuration is as follows:

- Frame 0: Non-banked, at address `0x0000 - 0x3FFF`
- Frame 1: `CODE_<N>`, at address `0x4000 - 0x7FFF`
  - Use for: Banked Code and Assets
  - Example: `#pragma codeseg CODE_2` or `#pragma codeseg CODE_255` for autobanking (no leading underscore)
  - Select the active bank using: `SWITCH_ROM()`. The current active bank can be queried using `CURRENT_BANK` or `MAP_FRAME1`
- Frame 2: `_LIT_<N>`, at address `0x8000- 0xBFFF`
  - Use for: Assets
  - `_DATA_N` may also be mapped into Frame 2 (RAM)
  - Example: `#pragma codeseg LIT_2` or `#pragma codeseg LIT_255` for autobanking (no leading underscore)
  - Select the active bank using `SWITCH_ROM2()`. The current active bank can be queried using `MAP_↔FRAME2`

Banked code and any pointers associated with it will only work correctly when active in Frame 1 (at `0x4000`), so it must use `CODE_<N>`. Graphics and other assets may go in either Frame 1 (at `0x4000`) or, if designed for it then Frame 2 (at `0x8000`).

### 6.7.1 Auto-Banking

`CODE` and `LIT` cannot share the same bank number. For example, if `CODE` is assigned to bank 3 then `LIT` cannot be in bank 3 as well.

`bankpack` is aware of this requirement and will group `CODE` and `LIT` separately when packing for autobanking. Its process is as follows:

1. Note: `CODE` and `LIT` are not sorted before packing
2. Assign fixed banks first (for both `CODE` and `LIT`). An error will be generated if both types assigned to the same bank. Banks are marked exclusive to whichever type is assigned in them first.
3. Then autobanked entries (both `CODE` and `LIT`) are assigned to banks, they are only assigned to banks of a matching type or an unused bank. Same as above, the first type to use a bank makes it exclusive to that type.

The `bankpack` option `-banktype=` may be used to set a bank to use specific type (`CODE` or `LIT`). This will take effect before `bankpack` tries to perform any bank assignment. For example: `-banktype=2:LIT` (or `-Wb-banktype=2:LIT` when used with `lcc`) sets bank 2 to exclusively use type `LIT`.

## 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 [sdlldb](#) (`lcc` equiv: `-Wl-y`)
  - `-j` enables `.noi` output for [sdlldb](#) (`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 sldlgb

The SDCC linker for the gameboy.

For detailed settings see [sldlgb-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 [sldlgb](#) 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 [sldlgb](#) 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:

- [gb\\_decompress\(\)](#), [gb\\_decompress\\_bkg\\_data\(\)](#), [gb\\_decompress\\_win\\_data\(\)](#), [gb\\_decompress\\_sprite\\_data\(\)](#)
- The `cross-platform/gbdecompress` example demonstrates how to use this compression

The utility can also compress (and decompress) using block style RLE encoding with the `--alg=rle` flag. Decompression support is available in GBDK:

- [rle\\_init\(\)](#), [rle\\_decompress\(\)](#)
- The `cross-platform/rle_map` example demonstrates how to use this compression

## 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 / Color, SGB borders, SMS/GG, NES

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.

For indexed color images, sometimes RGB paint programs mix up indexed colors in tiles if the same color exists in multiple palettes.

- `-repair_indexed_pal` can be used to fix this problem, though tiles must still follow the rule of using only one palette per tile.

Using this image a tileset 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](#)

## 7.6.4 png2hicolorgb

An updated version of Glen Cook's Windows GUI "hicolour.exe" 1.2 conversion tool for the Game Boy Color. The starting code base was the 1.2 release.

- For detailed settings see [Hi Color](#) on the Game Boy Color is a technique for displaying backgrounds with thousands of colors instead being limited to 32 colors for the entire screen background. It achieves this by changing ~16 colors of the background palette per scanline. The main tradeoffs are that it uses much of the Game Boy's available cpu processing per frame and requires more ROM space. The tile patterns, map, attributes and per-scanline palettes are pre-calculated using the PC based conversion tool.

For the current GBDK example ISR implementation there is a limit of 6 sprites per line before the hi-color timing breaks down and there start to be background artifacts.

Example: `png2hicolorgb myimage.png --csource -o=my_output_filename` Example with higher quality (slower conversion): `png2hicolorgb myimage.png --csource -o=my_output_filename --type=3 -L=2 -R=2`

Historical credits and info:

```
Original Concept : Icarus Productions
Original Code : Jeff Frohwein
Full Screen Modification : Anon
Adaptive Code : Glen Cook
Windows Interface : Glen Cook
Additional Windows Programming : Rob Jones
Original Quantiser Code : Benny
Quantiser Conversion : Glen Cook
```

**7.6.4.1 Additional Details** For technical details about the conversion process and rendering, see: <https://github.com/bbbbr/png2hicolorgb>

### 7.6.5 romusage

A utility for estimating usage of Game Boy and SMS/GG ROMs from .noi and .map files, binary ROMs and more.

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

Example: `romusage myprogram.noi -g`

## 8 Supported Consoles & Cross Compiling

### 8.1 Consoles Supported by GBDK

As of version 4.2.0 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)
- NES/Famicom (NES)
- MSX DOS (MSXDOS) (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. See below for available settings.



### 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 select the appropriate include path: `-I<gbdk-path>lib/<plat>`.

The assemblers used are:

- `sdasgb` (for GB/AP)
- `sdasz80` (for SMS/GG)
- `sdas6500` (for NES)

When linking:

- 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`

The linkers used are:

- `sldgb` (for GB/AP)
- `sldz80` (for SMS/GG or MSXDOS)
- `sld6808` (for NES)

MSXDOS requires an additional build step with `makecom` after `makebin` to create the final binary:

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

### 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`
- NES
  - `lcc: -mmos6502:nes`
  - `port:mos6502, plat:nes`
- MSX DOS
  - `lcc: -mz80:msxdos`
  - `port:z80, plat:msxdos`

## 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 `<nes/nes.h>` is included (either directly or through `<gbdk/platform.h>`)
  - `NINTENDO_NES` 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](#)
- [MAX\\_HARDWARE\\_SPRITES](#)
- [HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_X](#), [HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_Y](#)

## 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\\_bkg\\_native\\_data\(\)](#) to load the assets tiles in native format to the tile memory used for background tiles on that platform.

## 8.6 Hardware Summaries

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/DUCK

- 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 grid size: 8x8
  - tile attribute grid size: 8x8 (CGB mode only)
  - 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 grid size: 8x8
  - tile attribute grid size: 8x8
  - 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

#### NES/Famicom

- Sprites:
  - 8x8 or 8x16
  - 256 tiles
  - tile flipping/mirroring: yes
  - 64 total, max 8 per line
  - 4 x 4 color palette (color 0 transparent)
- Background: 256 tiles
  - tile grid size: 8x8
  - tile attribute grid size: 16x16 (bit packed into 32x32)
  - tile flipping/mirroring: no
  - 4 x 4 color palette (color 0 same for all sub-palettes)
- Window "layer": not available
- Screen: 256 x 240
- Hardware Map: Depends on mirroring mode
  - 256 x 240 (single-screen mirroring)
  - 512 x 240 (vertical mirroring / horizontal scrolling)
  - 256 x 480 (horizontal mirroring / vertical scrolling)
  - 512 x 480 (4-screen layout. Requires additional RAM on cartridge)

### 8.6.1 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.

#### NES/Famicom

- See [NES technical details](#)

## 8.7 Using Game Boy Color (GBC/CGB) Features

### 8.7.1 Differences Versus the Regular Game Boy (DMG/GBP/SGB)

These are some of the main hardware differences between the Regular Game Boy and the Game Boy Color.

- CPU: Optional 2x Speed mode
- Serial Link: Additional Speeds 2KB/s, 32KB/s, 64KB/s
- IR Port
- Sprites:
  - 2 banks x 256 tile patterns (2x as many) (typically upper 128 of each bank shared with background)
  - 8 x 4 color palettes in CGB mode (BGR-555 per color, 32768 color choices)
- Background:
  - 2 banks x 256 tile patterns (2x as many) (typically upper 128 of each bank shared with sprites)
  - Second map bank for tile attributes (color, flipping/mirroring, priority, bank)
  - 8 x 4 color palettes in CGB mode (BGR-555 per color, 32,768 color choices)
  - BG and Window master priority
- WRAM: 8 x 4K WRAM banks in the 0xD000 - 0xDFFF region
- LCD VRAM DMA

### 8.7.2 Game Boy Color features in GBDK

These are some of the main GBDK API features for the CGB. Many of the items listed below link to additional information.

- ROM header settings:
  - See the FAQ entry [How do I set SGB, Color only and Color compatibility in the ROM header?](#)
- Tile and Pattern data:
  - Select VRAM Banks: [VBK\\_REG](#) (used with `set_bkg/win/sprite_*`())

- [set\\_bkg\\_attributes\(\)](#), [set\\_bkg\\_submap\\_attributes\(\)](#)
- Color:
  - [set\\_bkg\\_palette\(\)](#), [set\\_bkg\\_palette\\_entry\(\)](#)
  - [set\\_sprite\\_palette\(\)](#), [set\\_sprite\\_palette\\_entry\(\)](#)
  - [set\\_default\\_palette\(\)](#)
  - [RGB\(\)](#), [RGB8\(\)](#), [RGBHTML\(\)](#)
- Detect and change CPU speed: if ([\\_cpu == CGB\\_TYPE](#)), [cpu\\_fast\(\)](#)
- More details in [cgb.h](#) (`#include <gb/cgb.h>`)

### 8.7.3 CGB Examples

Several examples in GBDK show how to use CGB features, including the following:

- [gb/colorbar](#), [gb/dscan](#), [cross-platform/large\\_map](#), [cross-platform/logo](#), [cross-platform/meta](#)

## 8.8 Porting Between Supported Consoles

### 8.8.1 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.

#### 8.8.1.1 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](#)
  - [LCDC](#) 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`

#### 8.8.1.2 Observed differences:

- MBC1 and MBC5 are supported, MBC3 won't save and RTC doesn't progress when game is not running, the HuC3 isn't supported at all (via JoseJX and sg).
- 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.

#### 8.8.1.3 Registers and Flags Use API defined registers and register flags instead of hardwired ones.

- LCDC register: [LCDC\\_REG](#) or [rLCDC](#)
- STAT register: [STAT\\_REG](#) or [rSTAT](#)
- LCDC flags: `-> LCDCF_...` (example: [LCDCF\\_ON](#))
- STAT flags: `-> STATF_...` (example: [STATF\\_LYC](#))

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

## 8.8.2 From Game Boy to SMS/GG

### 8.8.2.1 RAM Banks

- The SMS/GG ROM file size must be at least 64K to enable mapper support for RAM banks in emulators.
  - If the generated ROM is too small then `-yO 4` for makebin (or `-Wm-yO4` for LCC) can be used to set the size to 64K.

### 8.8.2.2 Tile Data and Tile Map loading

#### 8.8.2.2.1 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.

**8.8.2.2.2 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)

**8.8.2.3 Colors and Palettes** The SMS/GG have 2 x 16 color palettes:

- The first (0) is just for the background
- The second (1) is shared between sprites and the background (and for sprites a single color 0 of that palette is transparent)

On the Game Gear

- Each Palette is 32 bytes in size: 16 colors x 2 bytes per palette color entry.
- Each color (16 per palette) is packed as BGR-444 format (x:4:4:4, MSBits [15..12] are unused).
- Each component (R, G, B) may have values from 0 - 15 (4 bits), 15 is brightest.

On the SMS

- On SMS each Palette is 16 bytes in size: 16 colors x 1 byte per palette color entry.
- Each color (16 per palette) is packed as BGR-222 format (x:2:2:2, MSBits [7..6] are unused).

- Each component (R, G, B) may have values from 0 - 3 (2 bits), 3 is brightest.

For setting palette data:

- [set\\_palette\\_entry\(\)](#): Will set a single color in a palette
- [set\\_palette\(\)](#): Can set all the colors for one or both palettes
- [set\\_bkg\\_palette\(\)](#): Is just an alias for [set\\_palette\(\)](#). The full 16 colors can be set using this call.
- [set\\_sprite\\_palette\(\)](#): Is also an alias for [set\\_palette\(\)](#), but it offsets to write to the second 16 color palette.
- Also see: [RGB\(\)](#), [RGB8\(\)](#), [RGBHTML\(\)](#)

**8.8.2.3.1 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.

### 8.8.3 From Game Boy to NES

The NES graphics architecture is similar to the GB's. However, there are a number of design choices in the NES hardware that make the NES a particularly cumbersome platform to develop for, and that will require special attention.

Most notably:

- PPU memory can only be written in a serial fashion using a data port at 0x2007 (PPUDATA)
- PPU memory can only be written to during vblank, or when manually disabling rendering via PPUMASK. Hblank writes to PPU memory are not possible
- PPU memory write address is re-purposed for scrolling coordinates when rendering is enabled which means PPU memory updates / scrolling must cooperate
- PPU internal palette memory is also mapped to external VRAM area making palette updates during rendering very expensive and error-prone
- The base NES system has no support for any scanline interrupts. And cartridge mappers that add scanline interrupts do so using wildly varying solutions
- There's no easy way to determine the current scanline or CPU-to-PPU alignment meaning timed code is often required on the NES
- The PAL variant of the NES has very different CPU / PPU timings, as do the Dendy clone and other clone systems
- The stock 2 kB CPU RAM is just 1/4th the 8kB CPU RAM on a Game Boy
  - Free RAM after accounting for ZP, stack, OAM page and system variables further cuts this in half
  - This means a lot of GB code will need to be carefully optimized for RAM usage when ported to the NES
  - In particular, make sure to use the "const" modifier for arrays that are read-only, to make sure they don't end up in RAM

To provide an easier experience, gbdk-nes attempts to hide most of these quirks so that in theory the programming experience for gbdk-nes should be as close as possible to that of the GB/GBC. However, to avoid surprises it is recommended to familiarize yourself with the NES-specific quirks and implementation choices mentioned here.

This entire section is written as a guide on porting GB projects to NES. If you are new to GBDK, you may wish to familiarize yourself with using GBDK for GB development first as the topics covered will make a lot more sense after gaining experience with GB development.



**8.8.3.1 Mapper** Currently the NES support in GBDK uses UNROM-512 (Mapper30) with single-screen mirroring.

**8.8.3.2 Buffered mode vs direct mode** On the GB, the vblank period serves as an optimal time to write data to PPU memory, and PPU memory can also be written efficiently with VRAM DMA.

On the NES, writing PPU memory during the vblank period is not optional. Whenever rendering is turned on the PPU is in a state where accessing PPU memory results in undefined behavior outside the short vblank period. The NES also has no VRAM DMA hardware to help with data writes. This makes the vblank period not only more precious, but important to never exceed to avoid glitched games.

To deal with this limitation, all functions in gbdk-nes that write to PPU memory can run in either *Buffered* or *Direct* mode.

The good news is that switching between buffered and direct mode in gbdk-nes is usually done behind-the-scenes and normally shouldn't affect your code too much, as long as you use the portable GBDK functions and macros to do this.

- `DISPLAY_ON` / `SHOW_BG` / `SHOW_SPR` will all switch the system into buffered mode, allowing limited amounts of transfers during vblank, without affecting the display of graphics
- `DISPLAY_OFF` will switch the system into direct mode, allowing much larger/faster transfers while the screen is blanked

The following sections describe how the buffered / direct modes work in more detail. As buffered / direct mode is mostly hidden by the API calls, feel free to skip these sections if you wish.

**8.8.3.2.1 Buffered mode implementation details** To take maximum advantage of the short vblank period, gbdk-nes implements a popular optimization: An unrolled loop that pulls prepared data bytes from the stack.

```
PLA
STA PPUDATA
...
PLA
STA PPUDATA
RTS
```

The data structure to facilitate this is usually called a vram transfer buffer, often affectionately called a "popslide" buffer after Damian Yerrick's implementation. This buffer essentially forms a list of commands where each command sets up a new PPU address and then writes a sequence of bytes with an auto-increment of either +1 or +32. Each such command is often called a "stripe" in the nesdev community.

The transfer buffer starts at 0x100 and takes around half of the hardware stack page. You can think of the transfer buffer as a software-implemented DMA that allows writing bytes at the optimal rate of 8 cycles / byte. (ignoring the PPU address setup cost)

The buffer supports writing up to 32 continuous bytes at a time. This allows updating a full screen row / column, or two 8x8 tiles worth of tile data in one command / "stripe".

By doing writes to this buffer during game logic, your game will effectively keep writing data transfer commands for the vblank NMI handler to process in the next vblank period, without having to wait until the vblank.

Given that the transfer buffer only has space for around 100 data bytes, it is important to not overfill the buffer, as this will bring code execution to a screeching halt, until the NMI handler empties the old contents of the buffer to free up space and allow new commands to be written.

Buffered mode is typically used for scrolling updates or dynamically animated tiles, where only a small amount of bytes need updating per frame.

**8.8.3.2.2 Direct mode implementation details** During direct mode, all graphics routines will write directly to the PPUADDR / PPUDATA ports and the transfer buffer limit is never a concern because the transfer buffer is effectively bypassed.

Direct mode is typically used for initializing large amounts of tile data at boot and/or level loading time. Unless you plan to have an animated loading screen and decompress a lot of data, it makes more sense to just fade the screen to black and allow direct mode to write data as fast as possible.

**8.8.3.2.3 Caveat: Make sure the transfer buffer is emptied before switching to direct mode** Because the switch to direct mode is instant and doesn't wait for the next invocation of the vblank, it is possible to create situations where there is still remaining data in the transfer buffer that would only get written once the system is switched back to buffered mode.

To avoid this situation, make sure to always "drain" the buffer by doing a call to vsync when you expect your code to finish.

**8.8.3.2.4 Caveat: Only update the PPU palette during buffered mode** The oddity that PPU palette values are accessed through the same mechanism as other PPU memory bytes comes with the side effect that the vblank NMI handler will only write the palette values in buffered mode.

The reason for this design choice is two-fold:

- Having the NMI handler keep doing the palette updates when in direct mode would result in a race condition when the NMI handler interrupts the direct mode code and messes with the PPUADDR state that the direct mode code expects to remain unchanged
- Having the palette updates also switch to direct mode would run into another quirk of the system: Pointing PPUADDR at palette registers when display is turned off will make the display output that palette color instead of the common background color. The result would be glitchy artifacts on screen when updating the palette, leading to a slightly-glitchy looking game whenever the palette is updated with the screen off

To work around this, you are advised to never fully turn the display off during a palette fade. If you don't follow this advice all your palette updates will get delayed until the screen is turned back on.

**8.8.3.3 Shadow PPU registers** Like the SMS, the NES hardware is designed to only allow loading the full X/Y scroll on the very first scanline. i.e., under normal operation you are only allowed to change the Y-scroll once.

In contrast to the SMS, this limitation can be circumvented with a specific set of out-of-order writes to the PPUSCROLL/PPUADDR registers, taking advantage of the quirk that the PPUADDR and PPUSCROLL share register bits. But this write sequence is very timing-sensitive as the writes need to fall into (a smaller portion of) the hblank period in order to avoid race conditions when the CPU and PPU both try to update the same register during scanline rendering.

To simplify the programming interface, gbdk-nes functions like `move_bkg` / `scroll_bkg` only ever update shadow registers in RAM. The vblank NMI handler will later pick these values up and write them to the actual PPU registers.

**8.8.3.4 Implementation of (fake) vbl / lcd handlers** GBDK provides an API for installing Interrupt Service Routines that execute on start of vblank (VBL handler), or on a specific scanline (LCD handler).

But the base NES system has no suitable scanline interrupts that can provide such functionality. So instead, gbdk-nes API allows *fake* handlers to be installed in the goal of keeping code compatible with other platforms.

- An installed VBL handler will be called immediately when calling vsync. This handler should only update PPU shadow registers.
- An installed LCD handler for a specific scanline will then be called repeatedly until the value of `_lcd_scanline` is either set to an earlier scanline or  $\geq 240$ . After each invocation, shadow registers are stored into a buffer.
- After the vblank NMI handler has finished palette updates, OAM DMA, VRAM updates and scroll updates it will then manually run a delay loop to reach the particular scanlines that the installed LCD handler was pre-called for, and use the contents of the buffer to update registers.

Because the LCD "ISR" is actually implemented with a delay loop, it will burn a lot of CPU cycles in the frame - the further down the requested scanline is the larger the CPU cycle loss. In practice this makes this faked-LCD-ISR functionality mostly suitable for status bars at the top of the screen. Or for simple parallax cutscenes where the CPU has little else to do.

#### Note

The support for VBL and LCD handlers is currently under consideration and subject to change in newer versions of gbdk-nes.

**8.8.3.5 Caveat: Make sure to call vsync on every frame** On the GB, the call to vsync is an optional call that serves two purposes:

1. It provides a consistent frame timing for your game

2. It allows future register writes to be synchronized to the screen

On gbdk-nes the second point is no longer true, because writes need to be made to the shadow registers *before* vsync is called.

But the vsync call serves three other very important purposes:

A. It calls the optional VBL handler, where shadow registers can be written (and will later be picked up by the actual vblank NMI handler) B. It repeatedly calls the optional LCD handler up to MAX\_LCD\_ISR\_CALLS times. After each call, PPU shadow registers are stored into a buffer that will later be used by timed code in the NMI to handle mid-frame changes for screen splits / sprite hiding / etc. C. It calls flush\_shadow\_attributes so that updates to background attributes actually get written to PPU memory

For these reasons you should always include a call to vsync if you expect to see any graphical updates on the screen.

**8.8.3.6 Caveat: Do all status bar scroll movement in LCD handlers to mitigate glitches** The fake LCD ISR system is not bullet-proof. In particular, it has a problem where lag frames can cause the shadow register updates in LCD handlers not to be ready in time for when the timed code in the NMI handler would be called. This will effectively cause all those updates to be missing for one frame, and result in glitched scroll updates.

There is currently no complete work-around for this problem other than avoiding lag frames altogether. But the glitch can be made less distracting by making sure only the status bar glitches rather than the main background.

If you are using LCD handlers to achieve a top-screen stationary status bar, it is recommended that you follow the following guidelines to make sure the background itself has consistent scrolling:

- Use move\_bkg either in your main loop or in the VBL handler, to set the level scrolling
- Use move\_bkg in the first invocation of the LCD handler, to set the (stationary) status bar scroll position
- Use move\_bkg in the second invocation of the LCD handler, to reset the background scrolling

In short: Ensuring that the last called LCD handler sets the scroll back to the original value means the PPU rendering keeps rendering the background from the same scrolling position even when the NMI handling was missed.

### 8.8.3.7 Tile Data and Tile Map loading

#### 8.8.3.7.1 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 NES.
- [set\\_bkg\\_tiles\(\)](#) loads 1-byte-per-tile tilemaps both for the GB and NES.

**8.8.3.7.2 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
- NES: loads 2bpp tiles data

[set\\_tile\\_map\(\)](#)

- GB/AP: loads 1-byte-per-tile tilemaps
- NES: loads 1-byte-per-tile tilemaps

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](#)

Platform specific API calls:

- [set\\_bkg\\_attributes\\_nes16x16\(\)](#), [set\\_bkg\\_submap\\_attributes\\_nes16x16\(\)](#), [set\\_bkg\\_attribute\\_xy\\_nes16x16\(\)](#)

**8.8.3.7.3 Game Boy Color map attributes on the NES** 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 of setting `VBK_REG` to specify tile indices/attributes is not supported on the NES platform. Instead the dedicated functions for attribute setting should be used. These will work on other platforms as well and are the preferred way to set attributes.

To maintain API compatibility with other platforms that have attributes on an 8x8 grid specified with a whole byte per attribute, the NES platform supports the dedicated calls for setting attributes on an 8x8 grid:

- `set_bkg_attributes()`
- `set_bkg_submap_attributes()`
- `set_bkg_attribute_xy()`

This allows code to for attribute setting to remain unchanged between platforms. The effect of using these calls is that some attribute setting will be redundant due to the coarser attribute grid. i.e., setting the attribute at coordinates (4, 4), (4,5), (5, 4) and (5, 5) will all set the same attribute.

There is one more platform specific difference to note: While the `set_bkg_attribute_xy()` function takes coordinates on a 8x8 grid, the `set_bkg_attributes()` and `set_bkg_submap_attributes()` functions take a pointer to data in NES packed attribute format, where each byte contains data for 4 16x16 attribute. i.e. a 32x32 region.

While this implementation detail of how the attribute map is encoded is usually hidden by the API functions it does mean that code which manually tries to read the attribute data is *NOT* portable between NES/other platforms, and is not recommended.

#### Note

Tile map attributes on NES are on a 16x16 grid and use different control bits than the Game Boy Color.

- NES 16x16 Tile Attributes are bit packed into 4 attributes per byte with each 16x16 area of a 32x32 pixel block using the bits as follows:
  - D1-D0: Top-left 16x16 pixels
  - D3-D2: Top-right 16x16 pixels
  - D5-D4: Bottom-left 16x16 pixels
  - D7-D6: Bottom-right 16x16 pixels
  - [https://www.nesdev.org/wiki/PPU\\_attribute\\_tables](https://www.nesdev.org/wiki/PPU_attribute_tables)

### 8.8.4 From Game Boy to Mega Duck / Cougar Boy

The Mega Duck is (for practical purposes) functionally identical to the Original Game Boy though it has a couple changes listed below.

#### 8.8.4.1 Summary of Hardware changes:

- Cartridge Boot Logo: not present on Mega Duck
- Cartridge Header data: not present on Mega Duck
- Program Entry Point: `0x0000` (on Game Boy: `0x0100`)
- Display registers and flag definitions: Some changed
- Audio registers and flag definitions: Some changed
- MBC ROM bank switching register address: `0x0001` (many Game Boy MBCs use `0x2000` – `0x3FFF`)

**8.8.4.2 Best Practices** In order for software to be easily ported to the Mega Duck, or to run on both, use these practices. That will allow GBDK to automatically handle *most* of the differences (for the exceptions see [Sound Register Value Changes](#)).

- [Set the target console during build time](#)
- Use the GBDK definitions and macros for:

- Video Registers and Flags (examples: [LCD\\_C\\_REG](#), [LCD\\_C\\_F\\_BG8000](#), etc)
- Audio Registers and Flags (examples: [NR12\\_REG](#), [NR43\\_REG](#), etc)
- Use the default [SWITCH\\_ROM](#) macro for changing ROM banks

**8.8.4.3 Sound Register Value Changes** There are two hardware changes which will not be handled automatically when following the practices mentioned above.

These changes may be required when using existing Sound Effects and Music Drivers written for the Game Boy.

1. Registers [NR12\\_REG](#), [NR22\\_REG](#), [NR42\\_REG](#), and [NR43\\_REG](#) have their contents nybble swapped.
  - To maintain compatibility the value to write (or the value read) can be converted this way:  $((\text{uint8\_t})(\text{value} \ll 4) \mid (\text{uint8\_t})(\text{value} \gg 4))$
2. Register [NR32\\_REG](#) has the volume bit values changed.
  - Game Boy: Bits:6..5 : 00 = mute, 01 = 100%, 10 = 50%, 11 = 25%
  - Mega Duck: Bits:6..5 : 00 = mute, 01 = 25%, 10 = 50%, 11 = 100%
  - To maintain compatibility the value to write (or the value read) can be converted this way:  $((\sim(\text{uint8\_t})\text{value}) + (\text{uint8\_t})0x20u) \& (\text{uint8\_t})0x60u$

**8.8.4.4 Graphics Register Bit Changes** These changes are handled automatically when their GBDK definitions are used.

<a href="#">LCD_C_REG</a> Flag	Game Boy	Mega Duck		Purpose
<a href="#">LCD_C_F_B_ON</a>	.7	.7	(same)	Bit for LCD On/Off Select
<a href="#">LCD_C_F_B_WIN9C00</a>	.6	.3		Bit for Window Tile Map Region Select
<a href="#">LCD_C_F_B_WINON</a>	.5	.5	(same)	Bit for Window Display On/Off Control
<a href="#">LCD_C_F_B_BG8000</a>	.4	.4	(same)	Bit for BG & Window Tile Data Region Select
<a href="#">LCD_C_F_B_BG9C00</a>	.3	.2		Bit for BG Tile Map Region Select
<a href="#">LCD_C_F_B_OBJ16</a>	.2	.1		Bit for Sprites Size Select
<a href="#">LCD_C_F_B_OBJON</a>	.1	.0		Bit for Sprites Display Visible/Hidden Select
<a href="#">LCD_C_F_B_BGON</a>	.0	.6		Bit for Background Display Visible Hidden Select

**8.8.4.5 Detailed Register Address Changes** These changes are handled automatically when their GBDK definitions are used.

Register	Game Boy	Mega Duck
<a href="#">LCD_C_REG</a>	0xFF40	0xFF10
<a href="#">STAT_REG</a>	0xFF41	0xFF11
<a href="#">SCY_REG</a>	0xFF42	0xFF12
<a href="#">SCX_REG</a>	0xFF43	0xFF13
<a href="#">LY_REG</a>	0xFF44	0xFF18
<a href="#">LYC_REG</a>	0xFF45	0xFF19
<a href="#">DMA_REG</a>	0xFF46	0xFF1A
<a href="#">BGP_REG</a>	0xFF47	0xFF1B
<a href="#">OBP0_REG</a>	0xFF48	0xFF14
<a href="#">OBP1_REG</a>	0xFF49	0xFF15
<a href="#">WY_REG</a>	0xFF4A	0xFF16
<a href="#">WX_REG</a>	0xFF4B	0xFF17
-	-	-
<a href="#">NR10_REG</a>	0xFF10	0xFF20
<a href="#">NR11_REG</a>	0xFF11	0xFF22
<a href="#">NR12_REG</a>	0xFF12	0xFF21

Register	Game Boy	Mega Duck
<a href="#">NR13_REG</a>	0xFF13	0xFF23
<a href="#">NR14_REG</a>	0xFF14	0xFF24
-	-	-
<a href="#">NR21_REG</a>	0xFF16	0xFF25
<a href="#">NR22_REG</a>	0xFF17	0xFF27
<a href="#">NR23_REG</a>	0xFF18	0xFF28
<a href="#">NR24_REG</a>	0xFF19	0xFF29
-	-	-
<a href="#">NR30_REG</a>	0xFF1A	0xFF2A
<a href="#">NR31_REG</a>	0xFF1B	0xFF2B
<a href="#">NR32_REG</a>	0xFF1C	0xFF2C
<a href="#">NR33_REG</a>	0xFF1D	0xFF2E
<a href="#">NR34_REG</a>	0xFF1E	0xFF2D
-	-	-
<a href="#">NR41_REG</a>	0xFF20	0xFF40
<a href="#">NR42_REG</a>	0xFF21	0xFF42
<a href="#">NR43_REG</a>	0xFF22	0xFF41
<a href="#">NR44_REG</a>	0xFF23	0xFF43
-	-	-
<a href="#">NR50_REG</a>	0xFF24	0xFF44
<a href="#">NR51_REG</a>	0xFF25	0xFF46
<a href="#">NR52_REG</a>	0xFF26	0xFF45
-	-	-

## 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 rpn

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.
- Is it normal for sprites to disappear when they reach the left border of the screen? (NES/SMS/MSX)
  - You can hide the leftmost column using [HIDE\\_LEFT\\_COLUMN](#) to work around this.
  - The behavior is due to NES/SMS/MSX having 8-bit Sprite x coordinates while the screen width is also 256 pixels. GB/GG don't have this problem since their screen is smaller and the x-coordinates are larger than the visible screen.

## 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 Editors

- Why is VSCode flagging some GBDK types or functions as unidentified or giving warnings about them?
  - See [code\\_editors\\_hinting](#)
  - GBDK platform constants can be declared so that header files are parsed more completely in VSCode.

## 10.6 Errors and Warnings

- 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 the warning `?ASlink-Warning-Undefined Global ... mean?`
  - The linker is unable to find a variable or function that is referenced by some part of the program. This usually means a required source file is not being passed to the linker stage. Make sure all of your project source files are getting compiled and passed to the linker.
- 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.  
A common problem is when there is too much data in ROM0 (the lower 16K unbanked region) and it spills over into ROM1 (the first upper 16K banked region). Make sure ROM0 has 16K or less in it.  
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 do these errors mean? `error: size of the buffer is too small error: ROM is too large for number of banks specified`
  - 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 does warning 283: function declarator with no prototype mean?
  - Function forward declarations and definitions which have no arguments should be changed from `func()` to `func(void)`.
  - In C `func()` and `func(void)` do not mean the same thing. `()` means any number of parameters, `(void)` means no parameters. For example if a function with no arguments is declared with `()` then there may not be an error or warning when mistakenly trying to pass arguments to it.
- What do these warnings mean? warning 126: unreachable code warning 110↵  
: conditional flow changed by optimizer: so said EVELYN the modified  
DOG
  - The compiler is indicating that some branches of the code will never get executed or will have no effect, and so have been removed during optimization. There may be a logical error, unnecessary logic or unused variables.
  - If this is due to a variable which gets modified outside of normal execution (such as during an interrupt) then the `volatile` keyword can be added to it's declaration. This removes some optimizations and tells the compiler the variable's value may change unexpectedly and cannot be predicted only based on the surrounding code.
- What does this warning mean? WARNING: overflow in implicit constant conversion
  - See [Constants, Signed-ness and Overflows](#)
- On macOS, what does ... developer cannot be verified, macOS cannot verify that this app is free from malware mean?
  - It does not mean that GBDK is malware. It just means the GBDK toolchain binaries are not signed by Apple, so it won't run them without an additional step.
  - For the workaround, see the [macOS unsigned binary workaround](#) for 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.

## 10.7 Debugging / Compiling / Toolchain

- 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`. Also see [tools\\_debug](#).
- Is it possible to generate a debug symbol file (`.sym`) compatible with an 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`
- 
- 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.
  - Known issue: SDCC may fail on Windows when [run from folder names with spaces on non-C drives](#).

## 10.8 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 4.3.0

- GBDK now requires ~SDCC 4.4.0 or higher with GBDK-2020 patches for the z80 and NES
- Changed to new calling convention for `printf()`, `sprintf()`, `abs()`
- Changed to new SDCC calling convention for `set_bkg_tile_xy()`, `set_win_tile_xy()`
- The SDCC object file format (`.o`, `.rel` files) changed from XL3 (24 bit addresses) to XL4 (32 bit addresses)
  - Bankpack now supports both
- Recommend using:
  - `CURRENT_BANK` instead of `_current_bank`
  - `BANKED` macro instead of `__banked`
- NES `set_sprite_palette()` now indexes from 0 . . 3 instead of 4 . . 7
- `png2asset`:
  - If using either `-bpp 1` or `-pack_mode 1bpp` then the other is auto-enabled
  - Significant bug fixes and changes, check to make sure output is as expected

#### 11.1.2 Porting to GBDK-2020 4.2.0

- GBDK now requires ~SDCC 4.3 or higher with GBDK-2020 patches for the z80 and NES
- The following new functions replace old ones:
  - While the old functions will continue to work for now, migration to new versions is strongly encouraged
  - `vsync()`: replaces `wait_vbl_done()`
  - `set_default_palette()`: replaces `cgb_compatibility()`
  - `move metasprite_flip()`: replaces `move metasprite_hflip()`
  - `move metasprite_flipx()`: replaces `move metasprite_vflip()`
  - `move metasprite_flipxy()`: replaces `move metasprite_hvflip()`
  - `move metasprite_ex()`: replaces `move metasprite()`
- The unused `-DINT_16_BITS` argument was removed from the default SDCC compiler and preprocessor arguments (used in pre-GBDK2020 `gbdk/include-gb/types.h`)
- Removed legacy MBC register definitions `.MBC1_ROM_PAGE` and `.MBC_ROM_PAGE`
- SMS/GG
  - Swapped A and B buttons to match game boy buttons

#### 11.1.3 Porting to GBDK-2020 4.1.1

- No significant changes required

### 11.1.4 Porting to GBDK-2020 4.1.0

- GBDK now requires SDCC 4.2 or higher with GBDK-2020 patches for 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 existing 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 there are existing functions written in ASM which **call other functions written in C** the callee C function should be declared `OLDCALL`.
  - Function pointer declarations should be checked to see if they need `OLDCALL` added to the declaration.
    - \* Example (add `OLDCALL` at the end)
    - \* FROM: `typedef void (*someFunc)(uint8_t, uint8_t);`
    - \* TO: `typedef void (*someFunc)(uint8_t, uint8_t) 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

### 11.1.5 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

### 11.1.6 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/>`

### 11.1.7 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`

### 11.1.8 Porting to GBDK-2020 4.0.3

- No significant changes required

### 11.1.9 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.

### 11.1.10 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 unitialized 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.

### 11.1.11 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

### 11.1.12 Porting to GBDK-2020 3.2

- No significant changes required

### 11.1.13 Porting to GBDK-2020 3.1.1

- No significant changes required

### 11.1.14 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>

### 11.1.15 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 4.3.0

~2024/05

- Includes SDCC version ~4.4.0 (14650) with GBDK-2020 patches for Z80 and NES
  - [Patched SDCC Builds](#) with support for Sega GG/SMS and the Nintendo NES are used.
  - See the [github workflow](#) for details.
- Added native GBDK build for Apple ARM cpus
- Known Issues
  - SDCC may fail on Windows when [run from folder names with spaces on non-C drives](#).
- Library
  - Added [get\\_system\(\)](#) which indicates system speed
    - \* [SYSTEM\\_60HZ](#), [SYSTEM\\_50HZ](#), [SYSTEM\\_BITS\\_DENDY](#), [SYSTEM\\_BITS\\_NTSC](#), [SYSTEM\\_BITS\\_PAL](#), [SYSTEM\\_NTSC](#)
  - Changed to new calling convention for [printf\(\)](#), [sprintf\(\)](#), [abs\(\)](#)
  - Changed [EMU\\_printf\(\)](#) to remove dependency on `stdio.h` added similar [EMU\\_fmtbuf\(\)](#)
  - Fixed [emu\\_debug.h](#) macros missing a trailing space
  - NES
    - \* Added PAL support (detects NTSC/PAL/Dendy on reset to adjust fake LCD ISR timings)
    - \* Added BCD support



- \* Added deferred hblank system for fake LCD ISRs, allowing for multiple splits per-frame
- \* NMI optimization: Only call delay routine for remaining-vblank and LCD handler execution if LCD handler present
- \* NMI optimization: Skip OAM DMA and delay routines when display is off, to increase unbuffered performance
- \* NMI optimization: Save 8 cycles of stripe setup cost in vram transfer buffer execution
- \* ZP memory optimization: Make gbdk-nes functions use its own dedicated overlay segment
- \* Fixed `_map_tile_offset` not being applied for `set_bkg_based_tiles()`
- \* Fixed VRAM transfer buffer bug (ensure stack page cleared on reset)
- \* Fixed support for 4-player controllers using `fourscore`
- \* Fixed `set_sprite_palette()` to index from 0 . . 3 instead of 4 . . 7
- \* Fixed `__move metasprite` to initialize Y index register to zero
- \* Fixed `waitpadup` to wait for button release instead of press
- \* Updated libc to latest from sdcc 4.4.0
- SMS/GG
  - \* Added `SHOW_SPRITES`, `HIDE_SPRITES` (no hiding mid-frame)
  - \* Added `S_BANK` tile attribute
  - \* Added 6 button controller support in `joypad()`
  - \* Added BCD support
  - \* Added ability to move VDP SAT and name table to other locations by writing to VDP R2 and VDP R5.
    - Set name table to 0x1800 and SAT to 0x1F00 by default to free up some sprite tile space
    - Added `R5_SAT_0x1F00` definition for the R5 value controlling SAT position in VRAM
  - \* Added `__WRITE_VDP_REG_UNSAFE()` VDP macro while interrupts are disabled (such as in ISR handlers)
  - \* Added Game Gear registers and definitions
    - `GG_STATE`: `GGSTATE_STT`, `GGSTATE_NJAP`, `GGSTATE_NNTS`
    - `GG_EXT_7BIT`
    - `GG_EXT_CTL`: `GGEXT_NINIT`
    - `GG_SIO_SEND`
    - `GG_SIO_RECV`
    - `GG_SIO_CTL`: `SIOCTL_TXFL`, `SIOCTL_RXRD`, `SIOCTL_FRER`, `SIOCTL_INT`, `SIOCTL_TON`, `SIOCTL_RON`, `SIOCTL_BS0`, `SIOCTL_BS1`
    - `GG_SOUND_PAN`: `SOUNDPAN_TN1R`, `SOUNDPAN_TN2R`, `SOUNDPAN_TN3R`, `SOUNDPAN_NOSR`, `SOUNDPAN_TN1L`, `SOUNDPAN_TN2L`, `SOUNDPAN_TN3L`, `SOUNDPAN_NOSL`
  - \* Optimized native tile data loading routines
  - \* Improved `VDP_WRITE_DATA` macro so it does not need `di/ei` protection
  - \* Improved palette initialization in `crt0`
  - \* Fixed tilemap wrapping over the low bound of the VDP name table
  - \* Fixed `scroll_sprite()`
  - \* Fixed missing `sms.h` in `sms/metaspites.h`
  - \* Fixed scroll glitch due to accessing VDP too fast
- Z80 shared SMS/GG/MSX
  - \* Added `DIV_REG` emulation for the Z80 systems, may be useful for seeding RNG
  - \* Changed VDP to reduce chances of dangerous ISR nesting (see SDCC issue <https://sourceforge.net/p/sdcc/bugs/3721/>)
  - \* Changed to allow nested locking of the shadow SAT copying on VBlank
  - \* Optimized `memcpy()` for larger amounts of data
  - \* Fixed `waitpad()`
  - \* Fixed return result of `"set_tile x, y"` family functions
  - \* Fixed to not allow interrupts to fire durint `crt0` initialization code

- MSXDOS
  - \* Fixed `.VDP_COLORDATA2` assembly definition
- Game Boy
  - \* Added HBlank copy routines: `hblank_copy_vram()`, `hblank_cpy_vram()`, `hblank_copy()`
  - \* Added `SCF_START`, `SCF_SOURCE`, `SCF_SPEED` aliases for SIO (Serial/Link port) control register control constants
  - \* Added clamping and changed to new SDCC calling convention for `set_bkg_tile_xy()`, `set_win_tile_xy()`
  - \* Added `S_BANK` tile attribute
  - \* Fixed 8-bit signed modulus
- MegaDuck
  - \* Fixed ROM bank switching on hardware when trying to enable or switch SRAM banks
- Toolchain / Utilities
  - Added `romusage` utility for viewing free/used ROM and RAM in compiled programs
  - `lcc`
    - \* Changed `-debug` to add the following flags: `-Wa-l -Wl-u -Wl-w`
  - `png2asset`
    - \* Added `-sprite_no_optimize`: sprite conversion will keep duplicate and empty sprite tiles
    - \* Added `-entity_tileset`: mark entity locations on maps during conversion with an entity tileset
    - \* Added `-rel_paths`: paths to tilesets are relative to the input file path
    - \* Changed `-keep_palette_order` to round up to at least one palette
    - \* Changed `-use_structs + -source_tileset` behavior
      - Point tile data to external source tileset
      - Add `extra_tiles` struct member pointing to map tiles not found in source tileset (null if none found)
    - \* Changed `-use_structs` to use designated initializers
    - \* Changed if using either `-bpp 1` or `-pack_mode 1bpp` then the other is auto-enabled
    - \* Fixed garbage in unused colors of palettes (set unused colors to black)
    - \* Fixed `-bin` mode not honoring `-tiles_only` and `-maps_only`
    - \* Fixed segfault and wrong data size for `-pack_mode sgb + -bin`
    - \* Fixed missing Palette ID in attributes for multi-palette SMS/GG backgrounds
    - \* Fixed not taking `-bpp` into account when converting metasprites and emitting `<symbol>_↵ tile_pals[]`
    - \* Fixed crash when filename for `-o` and `-c` not specified
    - \* Fixed some attributes missing for metasprite export
  - `makebin`
    - \* Fixed crash when using `-yS` (`-Wm-yS` with `lcc`)
  - `bankpack`
    - \* Added `-banktype=` to allow forcing a bank type to CODE or LIT before packing starts
    - \* Added support for XL4 (32 bit addresses) object file format (in addition to existing XL3)
    - \* Changed minimum bank for auto packing from 1 to 0 (required for the NES)
- Examples
  - Added HBlank copy example for GB/AP/Duck
  - Added Reading SNES joypads on NES example
  - Added Game Boy Printer example
  - Added Joypad testing example

- Added Display System example to demonstrate [get\\_system\(\)](#)
- Added Platformer example
- Added `GBDK_DEBUG` Makefile environment var for turning on/off debug flags
- Changed wav sample: play waveforms on the SMS/GG PSG
- Changed Random Number example: only call [initrand\(\)](#) once
- Changed Large Map: support modified initial camera position
- Changed all examples: use [BANKED](#) macro instead of `__banked`
  - \* Also change some to use [CURRENT\\_BANK](#) instead of `_current_bank`
- Fixes for SMS/GG: Fonts, Large Map, gbdecompress
- Fixed NES version of Text Scroller to have multiple splits as other platforms do
- Fixed Simple Physics: joypad input caching was wrong
- Fixed Banks Non-Intrinsic: mismatched SRAM banks for final calculation, improved naming
- Removed Analogue Pocket examples that were just duplicates of Game Boy ones
- Docs:
  - Fixed search where some exact matches didn't return a result
  - Various updates and improvements
  - Added more historical release notes

### 12.1.2 GBDK-2020 4.2.0

2023/08

- Includes SDCC version ~4.3 with GBDK-2020 patches for Z80 and NES
  - [Patched SDCC Builds](#) with support for Sega GG/SMS and the Nintendo NES are used. See the [github workflow](#) for details
- Known Issues
  - SDCC may fail on Windows when [run from folder names with spaces on non-C drives](#).
- Library
  - Added `NORETURN` definition (for `_Noreturn`)
  - Added: [set\\_bkg\\_attributes\(\)](#), [set\\_bkg\\_submap\\_attributes\(\)](#), [set\\_bkg\\_attribute\\_xy\(\)](#)
  - The following new functions replace old ones. The old functions will continue to work for now, but migration to new versions is strongly encouraged.
    - \* [vsync\(\)](#): replaces [wait\\_vbl\\_done\(\)](#)
    - \* [set\\_default\\_palette\(\)](#): replaces [cgb\\_compatibility\(\)](#)
  - metasprites: added metasprite functions which can set base sprite property parameter (`__current↔_base_prop`) for GB/GBC and NES
    - \* [move\\_metasprite\\_flipy\(\)](#): replaces [move\\_metasprite\\_hflip\(\)](#)
    - \* [move\\_metasprite\\_flipx\(\)](#): replaces [move\\_metasprite\\_vflip\(\)](#)
    - \* [move\\_metasprite\\_flipxy\(\)](#): replaces [move\\_metasprite\\_hvflip\(\)](#)
    - \* [move\\_metasprite\\_ex\(\)](#): (replaces [move\\_metasprite\(\)](#))
  - NES
    - \* Added support for much of the GBDK API
    - \* Banking support (library and sdcc toolchain)
    - \* Added [set\\_bkg\\_attributes\\_nes16x16\(\)](#), [set\\_bkg\\_submap\\_attributes\\_nes16x16\(\)](#), [set\\_bkg\\_attribute\\_xy\\_nes16x16\(\)](#)
  - SMS/GG
    - \* Swapped A and B buttons to match game boy buttons

- \* X coordinate metasprite clipping on the screen edges
- Game Boy
  - \* Minor crt0 optimizations
  - \* Faster `vmemcpy()`, `set_data()`, `get_data()`
  - \* Fixed `hide_sprites_range(39u, 40u)`; overflow shadow OAM
  - \* Increased `sgb_transfer()` maximum packet length to 7 x 16 bytes
  - \* Convert `gb_decompress` routines to the new calling convention
  - \* Convert `rle_decompress` routines to the new calling convention
  - \* Removed legacy MBC register definitions `.MBC1_ROM_PAGE` and `.MBC_ROM_PAGE`
  - \* Workaround for possible HALT bug in Crash Handler
- Refactored interrupts to use less space
- Toolchain / Utilities
  - Added `png2hicolorgb`
  - `lcc`
    - \* Fixed `--sdccbindir`
    - \* Removed the unused `-DINT_16_BITS` from the default SDCC compiler and preprocessor arguments
    - \* Improved improved Game Gear header compatibility (change header region code from 4 to 6)
  - `png2asset`
    - \* Added `-o` as a more standard version of the `-c` argument
    - \* Added `-repair_index_pal`: Tries to repair tile palettes for indexed color pngs (such as when RGB paint programs mix up indexed colors if the same color exists in multiple palettes). Implies `-keep_palette_order`
    - \* Added `-no_palettes`: Do not export palette data
    - \* Fixed support for indexed color pngs with less than 8 bits color depth
    - \* Fixed incorrect palettes when different colors have same luma value (use RGB values as less-significant bits)
    - \* Fixed `-keep_duplicate_tiles` not working with `-source_tileset`
    - \* Changed to use cross-platform constants for metasprite properties (`S_FLIPX`, `S_FLIPY` and `S_↔PAL`)
  - `makebin`
    - \* Warn if RAM banks specified and file size of ROM is less than the 64K required to enable them with in emulators
  - Added `sld6808` (for NES)
- Examples
  - Fixed `mkdir` broken in some `compile.bat` files (remove unsupported `-p` flag during bat file conversion)
  - Sound Test: Added MegaDuck support
  - Wav Playback: Improved support on AGB/AGS hardware
  - Metasprites: Added sub-palette switching for GBC and NES, software metasprite flipping for sms/gg
  - Large Map: Added color for supported platforms
  - LCD ISR Wobble: Improved interrupt flag settings
  - Added GB-Type example
  - Added Game Boy Color Hi-Color example using `png2hicolorgb`
- Docs:
  - Improved search to do partial matches instead of matching start of string only
  - Added SDAS assembler manual (`asmInk_manual.txt`)

- Added section on [NES support](#)
- Added [Using Game Boy Color Features](#)
- Updated [MegaDuck hardware documentation](#)
- Added [Banked Calling Convention](#)
- Added mention of [MAX\\_HARDWARE\\_SPRITES](#)

### 12.1.3 GBDK-2020 4.1.1

2022/11

- Includes SDCC version 13350 with GBDK-2020 patches for Z80
- Library
  - Fixed [RGB\(\)](#) and [RGB8\(\)](#) macros

### 12.1.4 GBDK-2020 4.1.0

2022/10

- Includes SDCC version 13350 with GBDK-2020 patches for Z80
- Known Issues
  - The `compile.bat` batch files for Windows use the an invalid `-p` option for `mkdir`
- 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 `LDCDF_BG8000` bit of `LDCD_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

### 12.1.5 GBDK-2020 4.0.6

2022/02

- Includes SDCC version 12539 with GBDK-2020 patches for Z80
- 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 [memcpy\(\)](#)
  - 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

### 12.1.6 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](#)
    - \* [LCDCF\\_WINON](#), [LCDCF\\_WINOFF](#), [LCDCF\\_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

### 12.1.7 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

### 12.1.8 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

### 12.1.9 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 [hramcpy\(\)](#) (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

### 12.1.10 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

### 12.1.11 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, `sldgb` 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

### 12.1.12 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

### 12.1.13 GBDK-2020 3.1.1

2020/05/17

- Fixed issues with `libgcc_s_dw2-1.dll`

### 12.1.14 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://github.com/untoxa/bgb_profiling_toolkit/blob/master/readme.md) <https://gbdev.gg8.se/forums/viewtopic.php?id=710>

### 12.1.15 GBDK-2020 3.0.1

2020/04/12

- Updated SDCC to v.4.0
- Updated LCC to work with the new compiler

### 12.1.16 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 `sda5gb` 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\_← REG=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 -W-asm=rgbds or -W-asm=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 GBKDIR=...`
  - 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 gbk 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 :). gbk autodetects where it is installed - no more environment variables. `cygwin1.dll` and make are no longer required - gbk 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 gbk3-2.92 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: gbk | `/usr/lib/gbk`
  - 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 gbk/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 gbk-support first, which will download all the rest of the code. Alternatively, look at gbk-support and gbk-lib at `cvs.gbk.sourceforge.net`



and sdcc at [cvs.sdcc.sourceforge.net](http://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](http://sourceforge.net). Thanks must go to David Pfeffer for gbdk's previous resting place, [www.gbdev.org](http://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](http://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 [mallocl\(\)](#) was added.

### 12.2.11 GBDK 2.0b11 (DOS binary only) - 24 November 1997

- Fixed another bug in code generation, that could happen when performing logical operations on 1-byte variables.

### 12.2.12 GBDK 2.0b10 (DOS binary only) - 6 November 1997

- Fixed a nasty bug in code generation, that could happen when performing arithmetic operations on 1-byte variables.
- Changed the name of some files of the gb-dtmf example so that it compiles on DOS.

### 12.2.13 GBDK 2.0b9 (DOS binary only)

- Several bug fixes in the compiler and in the libraries.

### 12.2.14 GBDK 2.0b8 (DOS binary only)

- Limited all file names to 8 characters to solve problems on DOS.
- Added communication routines that enable to send data through the link port of the GameBoy. Unfortunately, these routines do not always work; so use them with care until the next GBDK release.
- Added the comm.c example which illustrates how to use communication routines.
- It is possible to specify the name of the program (to be written in the image header) at link time using the -Wl-yn="XXX" flag (where X is the name of the program, which can contain up to 16 characters in quotes, including spaces; on Unix, depending on your shell, you must add backslashes before quotes and spaces like in -Wl-yn="My\ Game").
- Several bug fixes in the compiler.

### 12.2.15 GBDK 2.0b7 (DOS binary only)

- GBDK now uses a pre-release of lcc 4.1 (DOS binary only), that fixes a couple of problems in code generation.
- A couple of important points have been documented in the GBDK Programming Guidelines and Known Problems sections.
- Several improvements and optimizations to the code generator.

### 12.2.16 GBDK 2.0b6

- Added a peephole optimizer (with few rules at the moment).
- Changed the name of the hardware registers to match the "official" names.
- Added support for copying complete functions to RAM or HIRAM ([memcpy\(\)](#) and [hramcpy\(\)](#) functions). The compiler now automatically generates two symbol for the start and the end of each function, named start\_X and end\_X (where X is the name of the function). This enables to calculate the length of a function when copying it to RAM.
- Added the ram\_fn.c example which illustrates how to copy functions to RAM and HIRAM.
- Added support for installing IRQ handlers.
- Added the irq.c example which illustrates how to install IRQ handlers.
- Added RAM banks support (switch\_ram\_bank() function). The switch\_bank() function has been renamed to switch\_rom\_bank(). The banks.c example has been updated. The flags for generating multiple bank images have been modified.
- It is possible to set the sprite ram location at link time using the -Wl-g.OAM=# flag (where # is the address of the sprite ram). The sprite ram address must begin at an address multiple of 0x100, and is 0xA0 bytes long.

### 12.2.17 GBDK 2.0b5

- New documentation (not finished yet).
- Fixed a bug that could generate wrong code in switch statements.
- Fixed a bug in int comparison.
- Added a DTMF program written by Osamu Ohashi.
- Added a game (Deep Scan) written by a friend of Osamu.
- Modified the [delay\(\)](#) function so that it takes a long parameter. It can be used to wait between 1 and 65536 milliseconds (0 = 65536). The pause() function has been removed.

### 12.2.18 GBDK 2.0b4

- Fixed a bug that could generate wrong code when using hexadecimal constants.
- A new example (galaxy.c) has been added. It is the C version of the space.s example. sprite.c has been removed.
- Most of the libraries have been split into small modules for reducing final code size.

### 12.2.19 GBDK 2.0b3

- GBDK can generate multiple-banks images, i.e. images greater than 32kB (see the banks example).
- It is possible to set the stack pointer at link time using the -Wl-g.STACK=# flag (where # is the address of the stack pointer). Several functions (e.g. show\_bkg()) have been changed into macros (e.g. SHOW\_BKG). The [delay\(\)](#) function waits exactly 1 millisecond, and the pause() waits 256 milliseconds. Linking with the standard libraries is no more required. The lib/gb.lib (lib\gb.lib on DOS) text file contains a list of modules in which to look for undefined symbols. The linker will parse this file, and link your code with the required modules only. The stdio library has been split in several object files, and only necessary modules will be added to your code, thus reducing its size. The GBDK distribution can be located anywhere in your system if you use the -Wo-lccdir=GBDK-DIR flag when invoking lcc. Bug fixes.

### 12.2.20 GBDK 2.0b2

- Lots of bug fixes.
- GBDK has to be in the \GBDK-2.0 directory on DOS machines.

### 12.2.21 GBDK 2.0b1

- The code generator has been completely rewritten with the new version of lcc. It produces much smaller and more efficient code. The size of the code is generally between 20 and 50% smaller. A number of small optimizations are still to be done.
- The size of basic types has been changed:
  - An int is 8 bits.
  - A long is 16 bits.
- This change was required for the code generator to produce better code, because the Z80 is actually an 8-bit processor.
- The linker generates the complement checksum correctly now.
- The libraries and example programs have been modified for the new code generator.

### 12.2.22 GBDK 1.1

- Removed Xloadimage from the GBDK distribution. It is now available as a separate archive.
- A compiled DOS version is now available (cross-compiled on my Sun Workstation!).
- The libraries and the example programs have been improved.
- The make script has been improved. Compiling on UNIX should be easier.
- Many bugfixes.

### 12.2.23 GBDK 1.0-1 1996

## 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), -j (.noi), -w (wide .map format) for linker
            -Wa-l (assembler .lst), -Wl-u (.lst -> .rst address update)
-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[pfabilm]arg pass 'arg' to the preprocessor, compiler, assembler, bankpack, linker, ihxcheck, or makebin
```

## 13.2 sdcc settings

SDCC : z80/sm83/mos6502/mos65c02 TD- 4.4.1 #14650 (Linux)

published under GNU General Public License (GPL)

Usage : sdcc [options] filename

Options :-

General options:

--help	Display this help
-v --version	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
-E --preprocessonly	Preprocess only, do not compile
--syntax-only	Parse and verify syntax only, do not compile
-S	Compile only; do not assemble or link
-c --compile-only	Compile and assemble, but do not link
--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 Micro\$oft 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	Determine the language standard (c89, c99, c11, c23, sdcc89 etc.)
--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-peeph-comments	don't include peephole optimizer comments
--codeseq	<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:

--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
--no-reg-params	On some ports, disable passing some parameters in registers
--nostdlibcall	Disable optimization of calls to standard library
--nooverlay	Disable overlaying leaf function auto variables
--nogcse	Disable the GCSE optimisation
--nolospre	Disable lospre
--nogenconstprop	Disable generalized constant propagation
--nolabelopt	Disable label optimisation
--noinvariant	Disable optimisation of invariants
--noinduction	Disable loop variable induction
--noloopreverse	Disable the loop reverse optimisation
--no-peeph	Disable the peephole assembly file optimisation
--peeph-asm	Enable peephole optimization on inline assembly
--peeph-return	Enable peephole optimization for return instructions
--no-peeph-return	Disable peephole optimization for return instructions
--peeph-file	<file> use this extra peephole file
--allow-unsafe-read	Allow optimizations to read any memory location anytime

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
--nmos-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-zp-spill       place register spills in 16-bit address space
--no-std-crt0       Do not link default crt0.rel
Special options for the mos65c02 port:
--model-small      8-bit address space for data
--model-large      16-bit address space for data (default)
--no-zp-spill       place register spills in 16-bit address space
--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 (Zilog Z80 / Hitachi HD64180 / ZX-Next / eZ80 / R800)
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 sdas6500 settings

```

sdas Assembler V02.00 + NoICE + SDCC mods (Rockwell 6502/6510/65C02)
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

## 13.6 backpack 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
-banktype=<b:t> : Set bank B (decimal) to use type T (CODE or LIT). For sms/gg
               Ex: -banktype=2:LIT sets bank 2 to type LIT
-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>;
```

## 13.7 sldlgb settings

sldl Linker V03.00/V05.40 + 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
-a (platform) Select platform specific virtual address translation
```

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 sldlz80 settings

sldl Linker V03.00/V05.40 + 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
-a    (platform) Select platform specific virtual address translation
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

```

### 13.9 sdld6808 settings

```

sdld Linker V03.00/V05.40 + sdld
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
-a    (platform) Select platform specific virtual address translation
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

```

### 13.10 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"

```

### 13.11 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             header rom size (0xa-0x2) (default: 0xc)
-xj n             set region code (3-7) (default: 4)
-xv n             version number (0-15) (default: 0)
-yo n             number of rom banks (default: 2) (autosize: A)
-ya n             number of ram banks (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)

```

## 13.12 makecom settings

makecom image.rom image.noi output.com

Use: convert a binary .rom file to .msxdos com format.

## 13.13 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

## 13.14 png2asset settings

```

usage: png2asset <file>.png [options]
-o <filename>      ouput file (if not used then default is <png file>.c)
-c <filename>      deprecated, same as -o
-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
-sprite_no_optimize keep empty sprite tiles, do not remove duplicate tiles
-b <bank>          bank (default: fixed bank)
-keep_palette_order use png palette
-repair_indexed_pal try to repair indexed tile palettes (implies "-keep_palette_order")
-noflip            disable tile flip
-map              Export as map (tileset + bg) instead of default metasprite output
-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. using 1 auto-enables "-pack_mode 1bpp")
-max_palettes       max number of palettes allowed (default: 8)
                   (note: max colors = max_palettes x num colors per palette)
-pack_mode          gb, nes, sgb, sms, 1bpp (default: gb. using 1bpp auto-enables "-bpp 1")
-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)
-entity_tileset     (maps only) mark matching tiles counting from 255 down, entity patterns not exported
-keep_duplicate_tiles do not remove duplicate tiles (default: not enabled)
-no_palettes        do not export palette data
-bin               export to binary format
-transposed         export transposed (column-by-column instead of row-by-row)
-rel_paths          paths to tilesets are relative to the input file path
decoder error empty input buffer given to decoder. Maybe caused by non-existing file?

```

## 13.15 png2hicolorgb settings

```

png2hicolorgb input_image.png [options]
version 1.4.1: bbbbr. Based on Glen Cook's Windows GUI "hicolour.exe" 1.2
Convert an image to Game Boy Hi-Color format
Options
-h          : Show this help
-v*         : Set log level: "-v" verbose, "-vQ" quiet, "-vE" only errors, "-vD" debug
-o <file>   : Set base output filename (otherwise from input image)
--csource   : Export C source format with incbins for data files
--bank=N    : Set bank number for C source output where N is decimal bank number 1-511
--type=N    : Set conversion type where N is one of below
              1: Median Cut - No Dither (*Default*)
              2: Median Cut - With Dither
              3: Wu Quantiser
-p          : Show screen attribute pattern options (no processing)
-L=N        : Set Left screen attribute pattern where N is decimal entry (-p to show patterns)
-R=N        : Set Right screen attribute pattern where N is decimal entry (-p to show patterns)
--vaddrid   : Map uses vram id (128->255->0->127) instead of (*Default*) sequential tile order (0->255)
--nodedupe  : Turn off tile pattern deduplication
Example 1: "png2hicolorgb myimage.png"
Example 2: "png2hicolorgb myimage.png --csource -o=my_output_filename"
* Default settings provide good results. Better quality but slower: "--type=3 -L=2 -R=2"
Historical credits and info:
  Original Concept : Icarus Productions
  Original Code   : Jeff Frohwein
  Full Screen Modification : Anon
  Adaptive Code   : Glen Cook
  Windows Interface : Glen Cook
  Additional Windows Programming : Rob Jones
  Original Quantiser Code : Benny
  Quantiser Conversion : Glen Cook

```

## 13.16 romusage settings

```

romusage input_file.[map|noi|ihx|cdb|.gb[c]|.pocket|.duck|.gg|.sms] [options]
version 1.2.8, by bbbbr
Options
-h : Show this help
-p:SMS_GG : Set platform to GBDK SMS/Game Gear (changes memory map templates)
-a : Show Areas in each Bank. Optional sort by, address:"-aA" or size:"-aS"
-g : Show a small usage graph per bank (-gA for ascii style)
-G : Show a large usage graph per bank (-GA for ascii style)
-B : Brief (summarized) output for banked regions. Auto scales max bank
    shows [Region]_[Max Used Bank] / [auto-sized Max Bank Num]
-F : Force Max ROM and SRAM bank num for -B. (0 based) -F:ROM:SRAM (ex: -F:255:15)
-m : Manually specify an Area -m:NAME:HEXADDR:HEXLENGTH
-e : Manually specify an Area that should not overlap -e:NAME:HEXADDR:HEXLENGTH
-E : All areas are exclusive (except HEADERS), warn for any overlaps
-q : Quiet, no output except warnings and errors
-Q : Suppress output of warnings and errors
-R : Return error code for Area warnings and errors
-sR : [Rainbow] Color output (-sRe for Row Ends, -sRd for Center Dimmed, -sRp % based)
-sP : Custom Color Palette. Colon separated entries are decimal VT100 color codes
      -sP:DEFAULT:ROM:VRAM:SRAM:WRAM:HRAM (section based color only)
-sC : Show Compact Output, hide non-essential columns
-sH : Show HEADER Areas (normally hidden)
-smROM : Show Merged ROM_0 and ROM_1 output (i.e. bare 32K ROM)
-smWRAM : Show Merged WRAM_0 and WRAM_1 output (i.e. DMG/MGB not CGB)
        -sm* compatible with banked ROM_x or WRAM_x when used with -B
-sJ : Show JSON output. Some options not applicable. When used, -Q recommended
-nB : Hide warning banner (for .cdb output)
-nA : Hide areas (shown by default in .cdb output)
-z : Hide areas smaller than SIZE -z:DECSIZE
Use: Read a .map, .noi, .cdb or .ihx file to display area sizes
Example 1: "romusage build/MyProject.map"
Example 2: "romusage build/MyProject.noi -a -e:STACK:DEFF:100 -e:SHADOW_OAM:C000:A0"
Example 3: "romusage build/MyProject.ihx -g"
Example 4: "romusage build/MyProject.map -q -R"
Example 5: "romusage build/MyProject.noi -sR -sP:90:32:90:35:33:36"
Example 6: "romusage build/MyProject.map -sRp -g -B -F:255:15 -smROM -smWRAM"
Notes:
  * GBDK / RGBDS map file format detection is automatic.

```

- \* Estimates are as close as possible, but may not be complete.  
Unless specified with -m/-e they \*do not\* factor regions lacking complete ranges in the Map/Noi/Ihx file, for example Shadow OAM and Stack.
- \* IHX files can only detect overlaps, not detect memory region overflows.
- \* CDB file output ONLY counts (most) data from C sources.  
It cannot count functions and data from ASM and LIBs,  
so bank totals may be incorrect/missing.
- \* GB/GBC/ROM files are just guessing, no promises.

## 14 Todo List

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

## 15 Module Index

### 15.1 C modules

Here is a list of all modules:

[List of gbdk fonts](#) 88

## 16 Data Structure Index

### 16.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">__far_ptr</a>	88
<a href="#">_fixed</a>	89
<a href="#">atomic_flag</a>	90
<a href="#">isr_nested_vector_t</a>	90
<a href="#">isr_vector_t</a>	90
<a href="#">joypads_t</a>	91
<a href="#">metasprite_t</a>	92
<a href="#">OAM_item_t</a>	93
<a href="#">sfont_handle</a>	94

## 17 File Index

### 17.1 File List

Here is a list of all files with brief descriptions:

<a href="#">gbdk-lib/include/assert.h</a>	116
<a href="#">gbdk-lib/include/ctype.h</a>	116
<a href="#">gbdk-lib/include/limits.h</a>	275

gbdk-lib/include/ <a href="#">rand.h</a>	341
gbdk-lib/include/ <a href="#">setjmp.h</a>	343
gbdk-lib/include/ <a href="#">stdarg.h</a>	98
gbdk-lib/include/ <a href="#">stdatomic.h</a>	371
gbdk-lib/include/ <a href="#">stdbool.h</a>	372
gbdk-lib/include/ <a href="#">stddef.h</a>	372
gbdk-lib/include/ <a href="#">stdint.h</a>	373
gbdk-lib/include/ <a href="#">stdio.h</a>	379
gbdk-lib/include/ <a href="#">stdlib.h</a>	380
gbdk-lib/include/ <a href="#">stdnoreturn.h</a>	384
gbdk-lib/include/ <a href="#">string.h</a>	110
gbdk-lib/include/ <a href="#">time.h</a>	384
gbdk-lib/include/ <a href="#">typeof.h</a>	385
gbdk-lib/include/ <a href="#">types.h</a>	115
gbdk-lib/include/asm/ <a href="#">types.h</a>	112
gbdk-lib/include/asm/mos6502/ <a href="#">provides.h</a>	95
gbdk-lib/include/asm/mos6502/ <a href="#">stdarg.h</a>	96
gbdk-lib/include/asm/mos6502/ <a href="#">string.h</a>	98
gbdk-lib/include/asm/mos6502/ <a href="#">types.h</a>	110
gbdk-lib/include/asm/sm83/ <a href="#">provides.h</a>	96
gbdk-lib/include/asm/sm83/ <a href="#">stdarg.h</a>	97
gbdk-lib/include/asm/sm83/ <a href="#">string.h</a>	102
gbdk-lib/include/asm/sm83/ <a href="#">types.h</a>	111
gbdk-lib/include/asm/z80/ <a href="#">provides.h</a>	96
gbdk-lib/include/asm/z80/ <a href="#">stdarg.h</a>	98
gbdk-lib/include/asm/z80/ <a href="#">string.h</a>	106
gbdk-lib/include/asm/z80/ <a href="#">types.h</a>	114
gbdk-lib/include/gb/ <a href="#">bcd.h</a>	118
gbdk-lib/include/gb/ <a href="#">bgb_emu.h</a>	124
gbdk-lib/include/gb/ <a href="#">cgb.h</a>	124
gbdk-lib/include/gb/ <a href="#">crash_handler.h</a>	130
gbdk-lib/include/gb/ <a href="#">drawing.h</a>	131

<a href="#">gbdk-lib/include/gb/emu_debug.h</a>	135
<a href="#">gbdk-lib/include/gb/gb.h</a>	139
<a href="#">gbdk-lib/include/gb/gbdecompress.h</a>	190
<a href="#">gbdk-lib/include/gb/hardware.h</a>	193
<a href="#">gbdk-lib/include/gb/hblankcpy.h</a>	240
<a href="#">gbdk-lib/include/gb/isr.h</a>	241
<a href="#">gbdk-lib/include/gb/metasprites.h</a>	243
<a href="#">gbdk-lib/include/gb/sgb.h</a>	263
<a href="#">gbdk-lib/include/gbdk/bcd.h</a>	120
<a href="#">gbdk-lib/include/gbdk/console.h</a>	266
<a href="#">gbdk-lib/include/gbdk/emu_debug.h</a>	136
<a href="#">gbdk-lib/include/gbdk/far_ptr.h</a>	267
<a href="#">gbdk-lib/include/gbdk/font.h</a>	270
<a href="#">gbdk-lib/include/gbdk/gbdecompress.h</a>	192
<a href="#">gbdk-lib/include/gbdk/gbdk-lib.h</a>	272
<a href="#">gbdk-lib/include/gbdk/incbin.h</a>	272
<a href="#">gbdk-lib/include/gbdk/metasprites.h</a>	250
<a href="#">gbdk-lib/include/gbdk/platform.h</a>	274
<a href="#">gbdk-lib/include/gbdk/rledecompress.h</a>	274
<a href="#">gbdk-lib/include/gbdk/version.h</a>	275
<a href="#">gbdk-lib/include/msx/hardware.h</a>	216
<a href="#">gbdk-lib/include/msx/metasprites.h</a>	250
<a href="#">gbdk-lib/include/msx/msx.h</a>	277
<a href="#">gbdk-lib/include/nes/bcd.h</a>	121
<a href="#">gbdk-lib/include/nes/hardware.h</a>	223
<a href="#">gbdk-lib/include/nes/metasprites.h</a>	253
<a href="#">gbdk-lib/include/nes/nes.h</a>	303
<a href="#">gbdk-lib/include/nes/rgb_to_nes_macro.h</a>	341
<a href="#">gbdk-lib/include/sms/bcd.h</a>	122
<a href="#">gbdk-lib/include/sms/gbdecompress.h</a>	192
<a href="#">gbdk-lib/include/sms/hardware.h</a>	227
<a href="#">gbdk-lib/include/sms/metasprites.h</a>	258

## 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 []` [extern]

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 []` [extern]

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`)(void)  
    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** `struct { ... } __far_ptr::segofs`

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

**19.1.2.6** `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** `struct { ... }`

**19.2.2.4** `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** `union { ... }`

**19.6.2.8** `union { ... }`

**19.6.2.9** `union { ... }`

**19.6.2.10** `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 <a href="#">set_sprite_tile</a> ) |
| <i>prop</i> | OAM Property Flags (see <a href="#">set_sprite_prop</a> ) |

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 <a href="#">set_sprite_tile</a> ) |
| <i>prop</i> | OAM Property Flags (see <a href="#">set_sprite_prop</a> ) |

### 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 0`

### 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 0`

## 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>

### Macros

- #define [memcpy](#)(dst, src, n) [\\_\\_memcpy](#)(dst, src, n)

### 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 Macro Definition Documentation

**20.21.2.1 memcpy** `#define memcpy(  
    dst,  
    src,  
    n ) \_\_memcpy(dst, src, n)`

### 20.21.3 Function Documentation

**20.21.3.1 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*

**20.21.3.2 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**

**20.21.3.3** `__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 |

**20.21.3.4** `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

**20.21.3.5** `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                      |

**20.21.3.6** `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`

**20.21.3.7** `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**

**20.21.3.8 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.

**20.21.3.9 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 <b>s2</b> |

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

**20.21.3.10 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**

**20.21.3.11 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 <b>s2</b> |

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**

**20.21.3.12 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**

|           |                       |
|-----------|-----------------------|
| <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**

**20.22.2.8 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.

**20.22.2.9 strncat()** `char* strncat (`

```

char * s1,
const char * s2,
int n )

```

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

#### Parameters

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

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

|           |                                     |
|-----------|-------------------------------------|
| <i>s1</i> | First string to compare             |
| <i>s2</i> | Second string to compare            |
| <i>n</i>  | 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 <b>s2</b> |

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 <b>s2</b> |

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 <b>s2</b> |

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 `NORETURN`
- #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 OLDSCALL** #define `OLDSCALL`

**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 NORETURN** `#define NORETURN`

**20.27.2.7 NONBANKED** `#define NONBANKED`

**20.27.2.8 BANKED** `#define BANKED`

**20.27.2.9 CRITICAL** `#define CRITICAL`

**20.27.2.10 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

|   |                   |
|---|-------------------|
| c | 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 <b>sour</b>                    |

**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 <b>sour</b>                    |

**20.32.4.4 bcd2text()** `uint8_t bcd2text (`  
     `const BCD * bcd,`  
     `uint8_t tile_offset,`  
     `uint8_t * buffer )`

Convert a BCD number into an ascii (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/nes/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`

#### 20.34.1 Detailed Description

Support for working with BCD (Binary Coded Decimal)  
See the example BCD project for additional details.

#### 20.34.2 Macro Definition Documentation

**20.34.2.1 BCD\_HEX** `#define BCD_HEX(  
v ) ((BCD)(v))`

**20.34.2.2 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

#### 20.34.3 Typedef Documentation

**20.34.3.1 BCD** `typedef uint32_t BCD`

#### 20.34.4 Function Documentation

**20.34.4.1 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 |

**20.34.4.2 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 <b>sour</b>                    |

**20.34.4.3 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 <b>sour</b>                    |

**20.34.4.4 bcd2text()** `uint8_t bcd2text (`  
     `const BCD * bcd,`  
     `uint8_t tile_offset,`  
     `uint8_t * buffer )`

Convert a BCD number into an ascii (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](#).

## 20.35 gbdk-lib/include/sms/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*)
- void [bcd\\_add](#) ([BCD](#) \**sour*, const [BCD](#) \**value*)
- void [bcd\\_sub](#) ([BCD](#) \**sour*, const [BCD](#) \**value*)
- [uint8\\_t](#) [bcd2text](#) (const [BCD](#) \**bcd*, [uint8\\_t](#) *tile\_offset*, [uint8\\_t](#) \**buffer*)

### 20.35.1 Detailed Description

Support for working with BCD (Binary Coded Decimal)  
See the example BCD project for additional details.

### 20.35.2 Macro Definition Documentation

**20.35.2.1 BCD\_HEX** `#define BCD_HEX(  
    v ) ((BCD) (v))`

**20.35.2.2 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

### 20.35.3 Typedef Documentation

**20.35.3.1 BCD** typedef [uint32\\_t](#) [BCD](#)

### 20.35.4 Function Documentation

**20.35.4.1 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 |

**20.35.4.2 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 <b>sour</b>                    |

**20.35.4.3 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 <b>sour</b>                    |

**20.35.4.4 bcd2text()** `uint8_t bcd2text (`  
     `const BCD * bcd,`  
     `uint8_t tile_offset,`  
     `uint8_t * buffer )`

Convert a BCD number into an ascii (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](#).

## 20.36 gbdk-lib/include/gb/bgb\_emu.h File Reference

```
#include <gbdk/emu_debug.h>
```

### 20.36.1 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.

## 20.37 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) | ((uint16_t)((g) & 0x1f) << 5) | ((r) & 0x1f))`
- `#define RGB8(r, g, b) (((uint16_t)((b) >> 3) & 0x1f) << 10) | ((uint16_t)((g) >> 3) & 0x1f) << 5) | (((r) >> 3) & 0x1f)`
- `#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, const palette_color_t *rgb_data) OLDCALL`
- `void set_sprite_palette (uint8_t first_palette, uint8_t nb_palettes, const 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)`
- `void cpu_fast (void)`
- `void set_default_palette (void)`
- `void cgb_compatibility (void)`

### 20.37.1 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.

## 20.37.2 Macro Definition Documentation

**20.37.2.1 RGB** `#define RGB(`  
    `r,`  
    `g,`  
    `b ) ((uint16_t)((b & 0x1f) << 10) | ((uint16_t)((g & 0x1f) << 5)) | ((r) &`  
    `0x1f))`

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\(\)](#)

**20.37.2.2 RGB8** `#define RGB8(`  
    `r,`  
    `g,`  
    `b ) (((uint16_t)((b >> 3) & 0x1f) << 10) | ((uint16_t)((g >> 3) & 0x1f)`  
    `<< 5)) | ((r) >> 3) & 0x1f)`

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\(\)](#)

**20.37.2.3 RGBHTML** `#define RGBHTML(`  
    `RGB24bit ) (RGB8(((RGB24bit) >> 16) & 0xff), ((RGB24bit) >> 8) & 0xff, ((RGB24bit)`  
    `& 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\(\)](#)

**20.37.2.4 RGB\_RED** `#define RGB_RED RGB(31, 0, 0)`

Common colors based on the EGA default palette.

**20.37.2.5 RGB\_DARKRED** `#define RGB_DARKRED RGB(15, 0, 0)`

**20.37.2.6 RGB\_GREEN** `#define RGB_GREEN RGB( 0, 31, 0)`

**20.37.2.7 RGB\_DARKGREEN** `#define RGB_DARKGREEN RGB( 0, 15, 0)`

**20.37.2.8 RGB\_BLUE** `#define RGB_BLUE RGB( 0, 0, 31)`

**20.37.2.9 RGB\_DARKBLUE** `#define RGB_DARKBLUE RGB( 0, 0, 15)`

**20.37.2.10 RGB\_YELLOW** `#define RGB_YELLOW RGB(31, 31, 0)`

**20.37.2.11 RGB\_DARKYELLOW** `#define RGB_DARKYELLOW RGB(21, 21, 0)`

**20.37.2.12 RGB\_CYAN** `#define RGB_CYAN RGB( 0, 31, 31)`

**20.37.2.13 RGB\_AQUA** `#define RGB_AQUA RGB(28, 5, 22)`

**20.37.2.14 RGB\_PINK** `#define RGB_PINK RGB(31, 0, 31)`

**20.37.2.15 RGB\_PURPLE** `#define RGB_PURPLE RGB(21, 0, 21)`

**20.37.2.16 RGB\_BLACK** `#define RGB_BLACK RGB( 0, 0, 0)`

**20.37.2.17 RGB\_DARKGRAY** `#define RGB_DARKGRAY RGB(10, 10, 10)`

**20.37.2.18 RGB\_LIGHTGRAY** `#define RGB_LIGHTGRAY RGB(21, 21, 21)`

**20.37.2.19 RGB\_WHITE** `#define RGB_WHITE RGB(31, 31, 31)`

**20.37.2.20 RGB\_LIGHTFLESH** `#define RGB_LIGHTFLESH RGB(30, 20, 15)`

**20.37.2.21 RGB\_BROWN** `#define RGB_BROWN RGB(10, 10, 0)`

**20.37.2.22 RGB\_ORANGE** `#define RGB_ORANGE RGB(30, 20, 0)`

**20.37.2.23 RGB\_TEAL** `#define RGB_TEAL RGB(15, 15, 0)`

### 20.37.3 Typedef Documentation

**20.37.3.1 palette\_color\_t** `typedef uint16_t palette_color_t`  
16 bit color entry

### 20.37.4 Function Documentation

**20.37.4.1 set\_bkg\_palette()** `void set_bkg_palette (`  
    `uint8_t first_palette,`  
    `uint8_t nb_palettes,`  
    `const 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](#)

**20.37.4.2 set\_sprite\_palette()** `void set_sprite_palette (`  
    `uint8_t first_palette,`  
    `uint8_t nb_palettes,`  
    `const 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 <i>first_palette</i> ) |
| <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](#)

**20.37.4.3 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](#)

**20.37.4.4 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](#)

**20.37.4.5 `cpu_slow()`** `void cpu_slow (`  
`void )`

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\(\)](#)

**20.37.4.6 `cpu_fast()`** `void cpu_fast (`  
`void ) [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](#)

**20.37.4.7 `set_default_palette()`** `void set_default_palette (`  
`void )`

Sets CGB palette 0 to be 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.

**20.37.4.8 `cgb_compatibility()`** `void cgb_compatibility (`  
`void ) [inline]`

Obsolete. This function has been replaced by [set\\_default\\_palette\(\)](#), which has identical behavior.

## 20.38 `gbdk-lib/include/gb/crash_handler.h` File Reference

### Functions

- `void __HandleCrash (void)`



### 20.38.1 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.

### 20.38.2 Function Documentation

**20.38.2.1 `__HandleCrash()`** `void __HandleCrash (`  
`void )`

Display the crash dump screen.

See the intro for this file for more details.

## 20.39 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 `gprintrn` (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)
- 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**

### 20.39.1 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.

### 20.39.2 Macro Definition Documentation

**20.39.2.1 GRAPHICS\_WIDTH** `#define GRAPHICS_WIDTH 160`

Size of the screen in pixels

**20.39.2.2 GRAPHICS\_HEIGHT** `#define GRAPHICS_HEIGHT 144`

**20.39.2.3 SOLID** `#define SOLID 0x00 /* Overwrites the existing pixels */`

**20.39.2.4 OR** `#define OR 0x01 /* Performs a logical OR */`

**20.39.2.5 XOR** `#define XOR 0x02 /* Performs a logical XOR */`

**20.39.2.6 AND** `#define AND 0x03 /* Performs a logical AND */`

**20.39.2.7 WHITE** `#define WHITE 0`

Possible drawing colours

**20.39.2.8 LTGREY** `#define LTGREY 1`

**20.39.2.9 DKGREY** `#define DKGREY 2`

**20.39.2.10 BLACK** `#define BLACK 3`

**20.39.2.11 M\_NOFILL** `#define M_NOFILL 0`

Possible fill styles for `box()` and `circle()`

**20.39.2.12 M\_FILL** `#define M_FILL 1`

**20.39.2.13 SIGNED** `#define SIGNED 1`

Possible values for `signed_value` in [gprintln\(\)](#) and [gprintn\(\)](#)

**20.39.2.14 UNSIGNED** `#define UNSIGNED 0`**20.39.3 Function Documentation****20.39.3.1 gprint()** `void gprint (`  
`char * str )`

Print the string 'str' with no interpretation

See also

[gotogxy\(\)](#)

**20.39.3.2 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\(\)](#)

**20.39.3.3 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\(\)](#)

**20.39.3.4 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\(\)](#)

**20.39.3.5 plot()** `void plot (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t colour,`  
    `uint8_t mode )`

Old style plot - try [plot\\_point\(\)](#)

**20.39.3.6 plot\_point()** `void plot_point (`  
    `uint8_t x,`  
    `uint8_t y )`

Plot a point in the current drawing mode and colour at **x,y**

**20.39.3.7 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.

**20.39.3.8 draw\_image()** `void draw_image (`  
    `uint8_t * data )`

Draw a full screen image at **data**

**20.39.3.9 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**

**20.39.3.10 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)

**20.39.3.11 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)

**20.39.3.12 getpix()** `uint8_t getpix (`  
`uint8_t x,`  
`uint8_t y )`

Returns the current colour of the pixel at **x,y**

**20.39.3.13 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\(\)](#)

**20.39.3.14 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\(\)](#)

**20.39.3.15 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 <a href="#">box()</a> , letter color with <a href="#">gprintf()</a> , etc).                                                                            |
| <i>backcolor</i> | Secondary or background color where applicable (fill color of rectangles with <a href="#">box()</a> when <a href="#">M_FILL</a> is specified, background color of text with <a href="#">gprintf()</a> , 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**

## 20.40 gbdk-lib/include/gb/emu\_debug.h File Reference

```
#include <gbdk/emu_debug.h>
```

### 20.40.1 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.

## 20.41 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↵  
text)
- #define [BGB\\_MESSAGE](#)(message\_text) [EMU\\_MESSAGE](#)(message\_text)
- #define [EMU\\_PROFILE\\_BEGIN](#)(MSG) [EMU\\_MESSAGE\\_SUFFIX](#)(MSG, "%ZEROCLKS%");
- #define [BGB\\_PROFILE\\_BEGIN](#)(MSG) [EMU\\_PROFILE\\_BEGIN](#)(MSG)
- #define [EMU\\_PROFILE\\_END](#)(MSG) [EMU\\_MESSAGE\\_SUFFIX](#)(MSG, "%-8+LASTCLKS%");
- #define [BGB\\_PROFILE\\_END](#)(MSG) [EMU\\_PROFILE\\_END](#)(MSG)
- #define [EMU\\_TEXT](#)(MSG) [EMU\\_MESSAGE](#)(MSG)
- #define [BGB\\_TEXT](#)(MSG) [EMU\\_TEXT](#)(MSG)
- #define [BGB\\_profiler\\_message](#) [EMU\\_profiler\\_message](#)()
- #define [BGB\\_printf](#)(...) [EMU\\_printf](#)(\_\_VA\_ARGS\_\_)
- #define [EMU\\_BREAKPOINT](#) \_\_asm\_\_("ld b, b");
- #define [BGB\\_BREAKPOINT](#) [EMU\\_BREAKPOINT](#)

### Functions

- void [EMU\\_profiler\\_message](#) (void)
- void [EMU\\_printf](#) (const char \*format,...) [PRESERVES\\_REGS](#)(a
- void [EMU\\_fmtbuf](#) (const unsigned char \*format, void \*data) [PRESERVES\\_REGS](#)(a

### Variables

- void [b](#)
- void [c](#)

#### 20.41.1 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)

#### 20.41.2 Macro Definition Documentation

**20.41.2.1 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

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| <i>message_text</i> | 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, LAST-CLKS, 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>

See also

```
EMU_PROFILE_BEGIN(), EMU_PROFILE_END()
```

**20.41.2.2 BGB\_MESSAGE** `#define BGB_MESSAGE(  
message_text ) EMU_MESSAGE(message_text)`

**20.41.2.3 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

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| <i>MSG</i> | 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()
```

**20.41.2.4 BGB\_PROFILE\_BEGIN** `#define BGB_PROFILE_BEGIN(  
MSG ) EMU_PROFILE_BEGIN(MSG)`

**20.41.2.5 EMU\_PROFILE\_END** `#define EMU_PROFILE_END(  
MSG ) EMU_MESSAGE_SUFFIX(MSG, "%-8+LASTCLKS%");`

Macro to **End** a profiling block and print the results in the emulator debug message window

Parameters

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| <i>MSG</i> | 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()
```

**20.41.2.6 BGB\_PROFILE\_END** `#define BGB_PROFILE_END(  
MSG ) EMU_PROFILE_END(MSG)`

**20.41.2.7 EMU\_TEXT** `#define EMU_TEXT(  
MSG ) EMU_MESSAGE(MSG)`

**20.41.2.8 BGB\_TEXT** `#define BGB_TEXT(  
MSG ) EMU_TEXT(MSG)`

**20.41.2.9 BGB\_profiler\_message** `#define BGB_profiler_message EMU_profiler_message()`

**20.41.2.10 BGB\_printf** `#define BGB_printf(  
... ) EMU_printf(__VA_ARGS__)`

**20.41.2.11 EMU\_BREAKPOINT** `#define EMU_BREAKPOINT __asm__("ld b, b");`  
The Emulator will break into debugger when encounters this line

**20.41.2.12 BGB\_BREAKPOINT** `#define BGB_BREAKPOINT EMU_BREAKPOINT`

### 20.41.3 Function Documentation

**20.41.3.1 EMU\_profiler\_message()** `void EMU_profiler_message (  
void )`

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%");`

**20.41.3.2 EMU\_printf()** `void EMU_printf (  
const char * format,  
... )`

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. 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.  
Currently supported in the Emulicious emulator



**20.41.3.3 EMU\_fmtbuf()** `void EMU_fmtbuf (`  
    `const unsigned char * format,`  
    `void * data )`

Print the string and arguments in the buffer buffer by the pointer given by format to the emulator debug message window

#### Parameters

|               |                                                      |
|---------------|------------------------------------------------------|
| <i>format</i> | The format string as per printf                      |
| <i>data</i>   | Buffer containing arguments, for example some struct |

#### See also

[EMU\\_printf](#) for the format string description

Currently supported in the Emulicious emulator

### 20.41.4 Variable Documentation

**20.41.4.1 b** `void b`

**20.41.4.2 c** `void c`

## 20.42 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` [SYSTEM\\_60HZ](#) 0x00
- `#define` [SYSTEM\\_50HZ](#) 0x01
- `#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\_BANK 0x08U
- #define S\_PALETTE 0x10U
- #define S\_FLIPX 0x20U
- #define S\_FLIPY 0x40U
- #define S\_PRIORITY 0x80U
- #define S\_PAL(n) n
- #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 DEVICE\_WINDOW\_PX\_OFFSET\_X
- #define MINWNDPOSY DEVICE\_WINDOW\_PX\_OFFSET\_Y
- #define MAXWNDPOSX (DEVICE\_WINDOW\_PX\_OFFSET\_X + DEVICE\_SCREEN\_PX\_WIDTH - 1)
- #define MAXWNDPOSY (DEVICE\_WINDOW\_PX\_OFFSET\_Y + DEVICE\_SCREEN\_PX\_HEIGHT - 1)
- #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(b) (\_current\_bank = (b), rROMB0 = (b))
- #define SWITCH\_RAM(b) (rRAMB = (b))
- #define ENABLE\_RAM (rRAMG = 0x0A)
- #define DISABLE\_RAM (rRAMG = 0x00)
- #define SWITCH\_ROM\_MEGADUCK(b) SWITCH\_ROM(b)
- #define SWITCH\_ROM\_MBC1(b) SWITCH\_ROM(b)
- #define SWITCH\_RAM\_MBC1(b) SWITCH\_RAM(b)
- #define ENABLE\_RAM\_MBC1 ENABLE\_RAM
- #define DISABLE\_RAM\_MBC1 DISABLE\_RAM
- #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) (\_current\_bank = (b), rROMB1 = 0, rROMB0 = (b))
- #define SWITCH\_ROM\_MBC5\_8M(b) (rROMB1 = ((uint16\_t)(b) >> 8), rROMB0 = (b))
- #define SWITCH\_RAM\_MBC5(b) SWITCH\_RAM(b)
- #define ENABLE\_RAM\_MBC5 ENABLE\_RAM
- #define DISABLE\_RAM\_MBC5 DISABLE\_RAM

- `#define DISPLAY_ON LCD_C_REG|=LCD_CF_ON`
- `#define DISPLAY_OFF display_off();`
- `#define HIDE_LEFT_COLUMN`
- `#define SHOW_LEFT_COLUMN`
- `#define SET_BORDER_COLOR(C)`
- `#define SHOW_BKG LCD_C_REG|=LCD_CF_BGON`
- `#define HIDE_BKG LCD_C_REG&=~LCD_CF_BGON`
- `#define SHOW_WIN LCD_C_REG|=LCD_CF_WINON`
- `#define HIDE_WIN LCD_C_REG&=~LCD_CF_WINON`
- `#define SHOW_SPRITES LCD_C_REG|=LCD_CF_OBJON`
- `#define HIDE_SPRITES LCD_C_REG&=~LCD_CF_OBJON`
- `#define SPRITES_8x16 LCD_C_REG|=LCD_CF_OBJ16`
- `#define SPRITES_8x8 LCD_C_REG&=~LCD_CF_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_attribute_xy set_bkg_attribute_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 HARDWARE_SPRITE_CAN_FLIP_X 1`
- `#define HARDWARE_SPRITE_CAN_FLIP_Y 1`
- `#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)`
- `void remove_LCD (int_handler h)`
- `void remove_TIM (int_handler h)`
- `void remove_SIO (int_handler h)`
- `void remove_JOY (int_handler h)`
- `void add_VBL (int_handler h)`
- `void add_LCD (int_handler h)`
- `void add_TIM (int_handler h)`
- `void add_low_priority_TIM (int_handler h)`
- `void add_SIO (int_handler h)`
- `void add_JOY (int_handler h)`
- `void nowait_int_handler (void)`
- `void wait_int_handler (void)`
- `uint8_t cancel_pending_interrupts (void)`
- `void mode (uint8_t m)`
- `uint8_t get_mode (void) PRESERVES_REGS(b`
- `uint8_t get_system (void)`
- `void send_byte (void)`
- `void receive_byte (void)`

- void `delay` (uint16\_t d) PRESERVES\_REGS(h
- uint8\_t `joypad` (void) PRESERVES\_REGS(b
- uint8\_t `waitpad` (uint8\_t mask) PRESERVES\_REGS(b
- void `waitpadup` (void) 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` (void) PRESERVES\_REGS(a
- void `disable_interrupts` (void) PRESERVES\_REGS(a
- void `set_interrupts` (uint8\_t flags) PRESERVES\_REGS(b
- void `reset` (void)
- void `vsync` (void) PRESERVES\_REGS(b
- void `wait_vbl_done` (void) PRESERVES\_REGS(b
- void `display_off` (void) PRESERVES\_REGS(b
- void `refresh_OAM` (void) 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) 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_attributes` (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*tiles)
- 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 `set_bkg_submap_attributes` (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*map, uint8\_t map\_w)
- 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)
- uint8\_t \* `set_bkg_attribute_xy` (uint8\_t x, uint8\_t y, uint8\_t a)
- 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)
- 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` (int8\_t x, int8\_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, 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)
- void `get_data` (uint8\_t \*data, uint8\_t \*vram\_addr, uint16\_t len)
- void `vmemcpy` (uint8\_t \*dest, uint8\_t \*sour, uint16\_t len)
- 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 `set_bkg_native_data` (uint8\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data)
- void `set_sprite_native_data` (uint8\_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 `vmemset` (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 d
- uint8\_t e
- uint8\_t h
- uint8\_t l
- 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 b
- 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

### 20.42.1 Detailed Description

Gameboy specific functions.

### 20.42.2 Macro Definition Documentation

#### 20.42.2.1 NINTENDO `#define NINTENDO`

**20.42.2.2 SYSTEM\_60HZ** `#define SYSTEM_60HZ 0x00`

**20.42.2.3 SYSTEM\_50HZ** `#define SYSTEM_50HZ 0x01`

**20.42.2.4 GAMEBOY** `#define GAMEBOY`

**20.42.2.5 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](#)

**20.42.2.6 J\_DOWN** `#define J_DOWN 0x08U`

**20.42.2.7 J\_LEFT** `#define J_LEFT 0x02U`

**20.42.2.8 J\_RIGHT** `#define J_RIGHT 0x01U`

**20.42.2.9 J\_A** `#define J_A 0x10U`

**20.42.2.10 J\_B** `#define J_B 0x20U`

**20.42.2.11 J\_SELECT** `#define J_SELECT 0x40U`

**20.42.2.12 J\_START** `#define J_START 0x80U`

**20.42.2.13 M\_DRAWING** `#define M_DRAWING 0x01U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

**20.42.2.14 M\_TEXT\_OUT** `#define M_TEXT_OUT 0x02U`

**20.42.2.15 M\_TEXT\_INOUT** `#define M_TEXT_INOUT 0x03U`

**20.42.2.16 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\(\)](#)

**20.42.2.17 M\_NO\_INTERP** `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

**20.42.2.18 S\_BANK** `#define S_BANK 0x08U`

If this bit set clear, the tile from the second VRAM bank is used

See also

[set\\_sprite\\_prop\(\)](#)

**20.42.2.19 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\(\)](#).

**20.42.2.20 S\_FLIPX** `#define S_FLIPX 0x20U`

If set the sprite will be flipped horizontally.

See also

[set\\_sprite\\_prop\(\)](#)

**20.42.2.21 S\_FLIPY** `#define S_FLIPY 0x40U`

If set the sprite will be flipped vertically.

See also

[set\\_sprite\\_prop\(\)](#)

**20.42.2.22 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\(\)](#)

**20.42.2.23 S\_PAL** `#define S_PAL(  
n ) n`

Defines how palette number is encoded in OAM. Required for the png2asset tool's metasprite output.

**20.42.2.24 EMPTY\_IFLAG** `#define EMPTY_IFLAG 0x00U`  
Disable calling of interrupt service routines

**20.42.2.25 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](#)

**20.42.2.26 LCD\_IFLAG** `#define LCD_IFLAG 0x02U`  
LCD Interrupt when triggered by the STAT register.

See also

[set\\_interrupts\(\)](#),  
[add\\_LCD](#)

**20.42.2.27 TIM\_IFLAG** `#define TIM_IFLAG 0x04U`  
Timer Interrupt when the timer [TIMA\\_REG](#) overflows.

See also

[set\\_interrupts\(\)](#),  
[add\\_TIM](#)

**20.42.2.28 SIO\_IFLAG** `#define SIO_IFLAG 0x08U`  
Serial Link Interrupt occurs when the serial transfer has completed.

See also

[set\\_interrupts\(\)](#),  
[add\\_SIO](#)

**20.42.2.29 JOY\_IFLAG** `#define JOY_IFLAG 0x10U`  
Joypad Interrupt occurs on a transition of the keypad.

See also

[set\\_interrupts\(\)](#),  
[add\\_JOY](#)

**20.42.2.30 DMG\_BLACK** `#define DMG_BLACK 0x03`



**20.42.2.31 DMG\_DARK\_GRAY** `#define DMG_DARK_GRAY 0x02`

**20.42.2.32 DMG\_LITE\_GRAY** `#define DMG_LITE_GRAY 0x01`

**20.42.2.33 DMG\_WHITE** `#define DMG_WHITE 0x00`

**20.42.2.34 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](#)

**20.42.2.35 SCREENWIDTH** `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`

Width of the visible screen in pixels.

**20.42.2.36 SCREENHEIGHT** `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`

Height of the visible screen in pixels.

**20.42.2.37 MINWNDPOSX** `#define MINWNDPOSX DEVICE_WINDOW_PX_OFFSET_X`

The Minimum X position of the Window Layer (Left edge of screen)

See also

[move\\_win\(\)](#)

**20.42.2.38 MINWNDPOSY** `#define MINWNDPOSY DEVICE_WINDOW_PX_OFFSET_Y`

The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move\\_win\(\)](#)

**20.42.2.39 MAXWNDPOSX** `#define MAXWNDPOSX (DEVICE_WINDOW_PX_OFFSET_X + DEVICE_SCREEN_PX_WIDTH - 1)`

The Maximum X position of the Window Layer (Right edge of screen)

See also

[move\\_win\(\)](#)

**20.42.2.40 MAXWNDPOSY** `#define MAXWNDPOSY (DEVICE_WINDOW_PX_OFFSET_Y + DEVICE_SCREEN_PX_HEIGHT - 1)`

The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move\\_win\(\)](#)

**20.42.2.41 DMG\_TYPE** `#define DMG_TYPE 0x01`

Hardware Model: Original GB or Super GB.

See also

[\\_cpu](#)

**20.42.2.42 MGB\_TYPE** `#define MGB_TYPE 0xFF`

Hardware Model: Pocket GB or Super GB 2.

See also

[\\_cpu](#)

**20.42.2.43 CGB\_TYPE** `#define CGB_TYPE 0x11`

Hardware Model: Color GB.

See also

[\\_cpu](#)

**20.42.2.44 GBA\_NOT\_DETECTED** `#define GBA_NOT_DETECTED 0x00`

Hardware Model: DMG, CGB or MGB.

See also

[\\_cpu, \\_is\\_GBA](#)

**20.42.2.45 GBA\_DETECTED** `#define GBA_DETECTED 0x01`

Hardware Model: GBA.

See also

[\\_cpu, \\_is\\_GBA](#)

**20.42.2.46 DEVICE\_SUPPORTS\_COLOR** `#define DEVICE_SUPPORTS_COLOR ( _cpu == CGB_TYPE)`

Macro returns TRUE if device supports color

**20.42.2.47 IO\_IDLE** `#define IO_IDLE 0x00U`

Serial Link IO is completed

**20.42.2.48 IO\_SENDING** `#define IO_SENDING 0x01U`

Serial Link Sending data

**20.42.2.49 IO\_RECEIVING** `#define IO_RECEIVING 0x02U`

Serial Link Receiving data

**20.42.2.50 IO\_ERROR** `#define IO_ERROR 0x04U`

Serial Link Error

**20.42.2.51 CURRENT\_BANK** `#define CURRENT_BANK _current_bank`

**20.42.2.52 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\(\)](#)

**20.42.2.53 BANKREF** `#define BANKREF( VARNAME )`

**Value:**

```
void __func_ ## VARNAME(void) __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.

**20.42.2.54 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 <a href="#">BANKREF()</a> |
|----------------|----------------------------------------------------------|

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

#### See also

[BANKREF\(\)](#), [BANK\(\)](#)

**20.42.2.55 SWITCH\_ROM** `#define SWITCH_ROM(  
    b ) ( _current_bank = (b) , rROMB0 = (b) )`

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](#)

**20.42.2.56 SWITCH\_RAM** `#define SWITCH_RAM(  
    b ) ( rRAMB = (b) )`

Switches SRAM bank on MBC1 and other compatible MBCs

#### Parameters

|          |                        |
|----------|------------------------|
| <i>b</i> | SRAM bank to switch to |
|----------|------------------------|

Before switching SRAM banks enable it using [ENABLE\\_RAM](#)

#### See also

[SWITCH\\_RAM\\_MBC1](#), [SWITCH\\_RAM\\_MBC5](#)

**20.42.2.57 ENABLE\_RAM** `#define ENABLE_RAM ( rRAMG = 0x0A )`  
Enables SRAM on MBC1 and other compatible MBCs

**20.42.2.58 DISABLE\_RAM** `#define DISABLE_RAM (rRAMG = 0x00)`

Disables SRAM on MBC1 and other compatible MBCs

**20.42.2.59 SWITCH\_ROM\_MEGADUCK** `#define SWITCH_ROM_MEGADUCK(`

`b ) SWITCH_ROM(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) |
|----------|------------------------------------------------------|

**20.42.2.60 SWITCH\_ROM\_MBC1** `#define SWITCH_ROM_MBC1(`

`b ) SWITCH_ROM(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>

**20.42.2.61 SWITCH\_RAM\_MBC1** `#define SWITCH_RAM_MBC1(`

`b ) SWITCH_RAM(b)`

Switches SRAM bank on MBC1 and other compatible MBCs

**Parameters**

|          |                        |
|----------|------------------------|
| <i>b</i> | SRAM bank to switch to |
|----------|------------------------|

Before switching SRAM banks enable it using [ENABLE\\_RAM](#)

See also

[SWITCH\\_RAM](#), [SWITCH\\_RAM\\_MBC5](#)

**20.42.2.62 ENABLE\_RAM\_MBC1** `#define ENABLE_RAM_MBC1 ENABLE\_RAM`

Enables SRAM on MBC1

**20.42.2.63 DISABLE\_RAM\_MBC1** `#define DISABLE_RAM_MBC1 DISABLE\_RAM`

Disables SRAM on MBC1

**20.42.2.64 SWITCH\_16\_8\_MODE\_MBC1** `#define SWITCH_16_8_MODE_MBC1 (*(volatile uint8_t *)0x6000 = 0x00)`**20.42.2.65 SWITCH\_4\_32\_MODE\_MBC1** `#define SWITCH_4_32_MODE_MBC1 (*(volatile uint8_t *)0x6000 = 0x01)`**20.42.2.66 SWITCH\_ROM\_MBC5** `#define SWITCH_ROM_MBC5(`

`b ) (_current_bank = (b), rROMB1 = 0, rROMB0 = (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

**20.42.2.67 SWITCH\_ROM\_MBC5\_8M** `#define SWITCH_ROM_MBC5_8M(  
    b ) (rROMB1 = ((uint16_t)(b) >> 8), rROMB0 = (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

**20.42.2.68 SWITCH\_RAM\_MBC5** `#define SWITCH_RAM_MBC5(  
    b ) SWITCH_RAM(b)`

Switches SRAM bank on MBC5

## Parameters

|          |                        |
|----------|------------------------|
| <i>b</i> | SRAM bank to switch to |
|----------|------------------------|

Before switching SRAM banks enable it using [ENABLE\\_RAM](#)

**20.42.2.69 ENABLE\_RAM\_MBC5** `#define ENABLE_RAM_MBC5 ENABLE\_RAM`  
Enables SRAM on MBC5

**20.42.2.70 DISABLE\_RAM\_MBC5** `#define DISABLE_RAM_MBC5 DISABLE\_RAM`  
Disables SRAM on MBC5

**20.42.2.71 DISPLAY\_ON** `#define DISPLAY_ON LCDC\_REG=LCDCF\_ON`  
Turns the display back on.

See also

[display\\_off](#), [DISPLAY\\_OFF](#)

#### 20.42.2.72 **DISPLAY\_OFF** `#define DISPLAY_OFF display\_off\(\);`

Turns the display off

Waits until the VBL before turning the display off.

See also

[display\\_off](#), [DISPLAY\\_ON](#)

#### 20.42.2.73 **HIDE\_LEFT\_COLUMN** `#define HIDE_LEFT_COLUMN`

Does nothing for GB

#### 20.42.2.74 **SHOW\_LEFT\_COLUMN** `#define SHOW_LEFT_COLUMN`

Does nothing for GB

#### 20.42.2.75 **SET\_BORDER\_COLOR** `#define SET_BORDER_COLOR( C )`

Does nothing for GB

#### 20.42.2.76 **SHOW\_BKG** `#define SHOW_BKG LCDC\_REG|=LCDCF\_BGON`

Turns on the background layer. Sets bit 0 of the LCDC register to 1.

Doesn't work in CGB mode - the bit is reused to control sprite priority over background and window layers instead.

- If 1 (SHOW\_BKG), everything works as usual.
- If 0 (HIDE\_BKG), all sprites are always drawn over background and window, ignoring any other priority settings.

#### 20.42.2.77 **HIDE\_BKG** `#define HIDE_BKG LCDC\_REG&=~LCDCF\_BGON`

Turns off the background layer. Sets bit 0 of the LCDC register to 0.

Doesn't work in CGB mode - the bit is reused to control sprite priority over background and window layers instead.

- If 1 (SHOW\_BKG), everything works as usual.
- If 0 (HIDE\_BKG), all sprites are always drawn over background and window, ignoring any other priority settings.

#### 20.42.2.78 **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](#).

#### 20.42.2.79 **HIDE\_WIN** `#define HIDE_WIN LCDC\_REG&=~LCDCF\_WINON`

Turns off the window layer. Clears bit 5 of the LCDC register to 0.

#### 20.42.2.80 **SHOW\_SPRITES** `#define SHOW_SPRITES LCDC\_REG|=LCDCF\_OBJON`

Turns on the sprites layer. Sets bit 1 of the LCDC register to 1.

**20.42.2.81 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](#)

**20.42.2.82 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.

**20.42.2.83 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.

**20.42.2.84 COMPAT\_PALETTE** `#define COMPAT_PALETTE(  
C0,  
C1,  
C2,  
C3 ) ((uint8_t) (((C3) << 6) | ((C2) << 4) | ((C1) << 2) | (C0)))`

**20.42.2.85 set\_bkg\_2bpp\_data** `#define set_bkg_2bpp_data set_bkg_data`

**20.42.2.86 set\_tile\_map** `#define set_tile_map set_bkg_tiles`

**20.42.2.87 set\_tile\_submap** `#define set_tile_submap set_bkg_submap`

**20.42.2.88 set\_tile\_xy** `#define set_tile_xy set_bkg_tile_xy`

**20.42.2.89 set\_attribute\_xy** `#define set_attribute_xy set_bkg_attribute_xy`

**20.42.2.90 set\_sprite\_2bpp\_data** `#define set_sprite_2bpp_data set_sprite_data`

**20.42.2.91 DISABLE\_OAM\_DMA** `#define DISABLE_OAM_DMA __shadow_OAM_base = 0`

**20.42.2.92 DISABLE\_VBL\_TRANSFER** `#define DISABLE_VBL_TRANSFER DISABLE_OAM_DMA`

Disable OAM DMA copy each VBlank

**20.42.2.93 ENABLE\_OAM\_DMA** `#define ENABLE_OAM_DMA __shadow_OAM_base = (uint8_t) ((uint16_t) &shadow_OAM  
>> 8)`

**20.42.2.94 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

**20.42.2.95 MAX\_HARDWARE\_SPRITES** `#define MAX_HARDWARE_SPRITES 40`

Amount of hardware sprites in OAM



**20.42.2.96 `HARDWARE_SPRITE_CAN_FLIP_X`** `#define HARDWARE_SPRITE_CAN_FLIP_X 1`  
True if sprite hardware can flip sprites by X (horizontally)

**20.42.2.97 `HARDWARE_SPRITE_CAN_FLIP_Y`** `#define HARDWARE_SPRITE_CAN_FLIP_Y 1`  
True if sprite hardware can flip sprites by Y (vertically)

**20.42.2.98 `fill_rect`** `#define fill_rect fill_bkg_rect`

### 20.42.3 Typedef Documentation

**20.42.3.1 `int_handler`** `typedef void(* int_handler) (void) NONBANKED`  
Interrupt handlers

**20.42.3.2 `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 <a href="#">set_sprite_tile</a> ) |
| <i>prop</i> | OAM Property Flags (see <a href="#">set_sprite_prop</a> ) |

### 20.42.4 Function Documentation

**20.42.4.1 `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\(\)](#)

**20.42.4.2 `remove_LCD()`** `void remove_LCD (`  
    `int_handler h )`

Removes the LCD interrupt handler.

See also

[add\\_LCD\(\)](#), [remove\\_VBL\(\)](#)

**20.42.4.3 remove\_TIM()** `void remove_TIM (`  
`int_handler h )`

Removes the TIM interrupt handler.

See also

[add\\_TIM\(\)](#), [remove\\_VBL\(\)](#)

**20.42.4.4 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 ([send\\_byte\(\)](#), [receive\\_byte\(\)](#)).

Once installed the default SIO ISR cannot be removed. Only secondary chained SIO ISRs (added with [add\\_SIO\(\)](#)) can be removed.

**20.42.4.5 remove\_JOY()** `void remove_JOY (`  
`int_handler h )`

Removes the JOY interrupt handler.

See also

[add\\_JOY\(\)](#), [remove\\_VBL\(\)](#)

**20.42.4.6 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.

**20.42.4.7 add\_LCD()** `void add_LCD (`  
`int_handler h )`

Adds a LCD interrupt handler.

Called when the LCD interrupt occurs.

Up to 3 handlers may be added, with the last added being called last.

There are various sources controlled by the [STAT\\_REG](#) register (\$FF41) which can trigger this interrupt. Common examples include triggering on specific scanlines using [LY\\_REG](#) == [LYC\\_REG](#). Another is applying graphics effects on a per-scanline basis such as modifying the X and Y scroll registers ([SCX\\_REG](#) / [SCY\\_REG](#) registers).

#### Note

LYC may not trigger with scanline 0 in the same way as other scanlines due to particular behavior with scanlines 153 and 0. Instead, using an [add\\_VBL\(\)](#) interrupt handler for start of frame behavior may be more suitable.

**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](#)/LCD/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.

**20.42.4.8 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\(\)](#)

**20.42.4.9 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\(\)](#)

**20.42.4.10 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.

The default SIO ISR gets installed automatically if any of the standard SIO calls are used ([send\\_byte\(\)](#), [receive\\_byte\(\)](#)).

See also

[send\\_byte](#), [receive\\_byte\(\)](#), [add\\_VBL\(\)](#)  
[set\\_interrupts\(\)](#) with [SIO\\_IFLAG](#)

**20.42.4.11 add\_JOY()** `void add_JOY (`  
`int_handler h )`

Adds a joypad button change interrupt handler.

This interrupt occurs on a transition of any of the keypad input lines from high to low, if the relevant [P1\\_REG](#) bits 4 or 5 are set.

For details about configuring flags or reading the data see: [https://gbdev.io/pandocs/Interrupt\\_Sources.html#int-60-joypad-interrupt](https://gbdev.io/pandocs/Interrupt_Sources.html#int-60-joypad-interrupt) [https://gbdev.io/pandocs/Joyypad\\_Input.html#ff00-p1joyp-joypad](https://gbdev.io/pandocs/Joyypad_Input.html#ff00-p1joyp-joypad)

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.

An example use of this is allowing the user to trigger an exit from the lower-power STOP cpu state.

See also

[joypad\(\)](#), [add\\_VBL\(\)](#), [IEF\\_HILO](#), [P1F\\_5](#), [P1F\\_4](#), [P1F\\_3](#), [P1F\\_2](#), [P1F\\_1](#), [P1F\\_0](#), [P1F\\_GET\\_DPAD](#), [P1F\\_GET\\_BTN](#), [P1F\\_GET\\_NONE](#)

**20.42.4.12 nowait\_int\_handler()** `void nowait_int_handler (`  
`void )`

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\(\)](#)

**20.42.4.13 wait\_int\_handler()** `void wait_int_handler (`  
`void )`

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\(\)](#)

**20.42.4.14 cancel\_pending\_interrupts()** `uint8_t cancel_pending_interrupts (`  
`void ) [inline]`

Cancel pending interrupts

**20.42.4.15 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](#)

**20.42.4.16 get\_mode()** `uint8_t get_mode (`  
    `void )`

Returns the current mode

See also

[M\\_DRAWING](#), [M\\_TEXT\\_OUT](#), [M\\_TEXT\\_INOUT](#), [M\\_NO\\_SCROLL](#), [M\\_NO\\_INTERP](#)

**20.42.4.17 get\_system()** `uint8_t get_system (`  
    `void ) [inline]`

Returns the system gbdk is running on.

See also

[SYSTEM\\_50HZ](#), [SYSTEM\\_60HZ](#), [SYSTEM\\_BITS\\_DENDY](#), [SYSTEM\\_BITS\\_NTSC](#), [SYSTEM\\_BITS\\_PAL](#),  
[SYSTEM\\_NTSC](#) [SYSTEM\\_PAL](#)

**20.42.4.18 send\_byte()** `void send_byte (`  
    `void )`

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](#)

**20.42.4.19 receive\_byte()** `void receive_byte (`  
    `void )`

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](#)

**20.42.4.20 delay()** `void delay (`  
    `uint16_t d )`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

**20.42.4.21** `joypad()` `uint8_t` joypad (  
    `void` )

Reads and returns the current state of the joypad. Follows Nintendo's guidelines for reading the pad. Return value is an OR of J\_\*

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](#)

**20.42.4.22** `waitpad()` `uint8_t` waitpad (  
    `uint8_t` mask )

Waits until at least one of the buttons given in mask are pressed.

Parameters

|                   |                                              |
|-------------------|----------------------------------------------|
| <code>mask</code> | 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](#)

**20.42.4.23** `waitpadup()` `void` waitpadup (  
    `void` )

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.

**20.42.4.24** `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

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| <code>npads</code>   | number of joypads requested (1, 2 or 4)                          |
| <code>joypads</code> | pointer to <a href="#">joypads_t</a> 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](#)

**20.42.4.25 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 <a href="#">joypads_t</a> structure to be filled with joypad statuses, must be previously initialized with <a href="#">joypad_init()</a> |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|

See also

[joypad\\_init\(\)](#), [joypads\\_t](#)

**20.42.4.26 enable\_interrupts()** `void enable_interrupts (`  
     void ) [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](#)

**20.42.4.27 disable\_interrupts()** `void disable_interrupts (`  
     void ) [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](#)

**20.42.4.28 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](#)

**20.42.4.29 reset()** `void reset (`  
`void )`

Performs a soft reset.

For the Game Boy and related it does this by jumping to address 0x0150 which is in crt0.s (the c-runtime that executes before main() is called).

This performs various startup steps such as resetting the stack, clearing WRAM and OAM, resetting initialized variables and some display registers (scroll, window, LCDC), etc.

This is not the same a hard power reset.

**20.42.4.30 vsync()** `void vsync (`  
`void )`

HALTs the CPU and waits for the vertical blank interrupt and then returns when all registered VBL ISRs have completed.

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.

**20.42.4.31 wait\_vbl\_done()** `void wait_vbl_done (`  
`void )`

Obsolete. This function has been replaced by [vsync\(\)](#), which has identical behavior.

**20.42.4.32 display\_off()** `void display_off (`  
`void )`

Turns the display off.

Waits until the VBL before turning the display off.

**See also**

[DISPLAY\\_ON](#)

**20.42.4.33 refresh\_OAM()** `void refresh_OAM (`  
`void )`

Copies data from shadow OAM to OAM

**20.42.4.34 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.                          |



**20.42.4.35 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               |

**20.42.4.36 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

Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

**20.42.4.37 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

**20.42.4.38 set\_2bpp\_palette()** `void set_2bpp_palette (`  
    `uint16_t palette ) [inline]`

Sets palette for 2bpp color translation for GG/SMS, does nothing on GB

**20.42.4.39 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.

**20.42.4.40 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](#)

**20.42.4.41 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 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](#)

**20.42.4.42 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](#)

**20.42.4.43** **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 |

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.42.4.44** **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 |

## Parameters

|              |                                              |
|--------------|----------------------------------------------|
| <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](#)

**20.42.4.45 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

**20.42.4.46 set\_bkg\_attributes()** `void set_bkg_attributes (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t w,`  
`uint8_t h,`  
`const uint8_t * tiles ) [inline]`

Sets a rectangular region of Background Tile Map Attributes.

#### 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 attribute 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\\_attributes\(\)](#) 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.

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\\_attributes](#), [set\\_win\\_tiles](#), [set\\_tiles](#)

Note

On the Game Boy this is only usable in Game Boy Color mode

#### 20.42.4.47 `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> ↔<br><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](#)

#### 20.42.4.48 [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

#### 20.42.4.49 [set\\_bkg\\_submap\\_attributes\(\)](#)

```
void set_bkg_submap_attributes (
    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 Attributes using a sub-region from a source tile attribute 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 attribute data                                                                |
| <i>map</i> ↔<br><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\\_attributes](#), [set\\_win\\_submap](#), [set\\_tiles](#)

Note

On the Game Boy this is only usable in Game Boy Color mode

```
20.42.4.50 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 |
| <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                   |

#### Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.42.4.51 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

**20.42.4.52 set\_bkg\_attribute\_xy()** `uint8_t* set_bkg_attribute_xy (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t a ) [inline]`

Set single attribute data a on background layer at x,y

#### Parameters

|          |                 |
|----------|-----------------|
| <i>x</i> | X-coordinate    |
| <i>y</i> | Y-coordinate    |
| <i>a</i> | tile attributes |

#### Returns

returns the address of tile attribute, so you may use faster [set\\_vram\\_byte\(\)](#) later

#### Note

On the Game Boy this is only usable in Game Boy Color mode

**20.42.4.53 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

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

**20.42.4.54 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](#)

**20.42.4.55 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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i> | Number of pixels to move the Background on the <b>Y axis</b><br>Range: -128 - 127 |

## See also

[move\\_bkg](#)

**20.42.4.56 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

**20.42.4.57 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](#)

**20.42.4.58 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\\_win\\_data](#), [set\\_1bpp\\_colors](#)  
[set\\_bkg\\_1bpp\\_data](#), [set\\_sprite\\_1bpp\\_data](#)

**20.42.4.59 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 |

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.42.4.60 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](#)

**20.42.4.61 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

**20.42.4.62 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> ↔<br><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](#)

**20.42.4.63 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

**20.42.4.64 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               |

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.42.4.65** `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

**20.42.4.66** `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

|          |              |
|----------|--------------|
| <i>x</i> | X-coordinate |
| <i>y</i> | Y-coordinate |

## Returns

returns the tile index



## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

**20.42.4.67 move\_win()** `void move_win (`  
    `uint8_t x,`  
    `uint8_t y ) [inline]`

Moves the Window to the **x**, **y** position on the screen.

## Parameters

|          |                                                                                    |
|----------|------------------------------------------------------------------------------------|
| <i>x</i> | X coordinate for Left edge of the Window (actual displayed location will be X - 7) |
| <i>y</i> | 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](#)

**20.42.4.68 scroll\_win()** `void scroll_win (`  
    `int8_t x,`  
    `int8_t y ) [inline]`

Move the Window relative to its current position.

## Parameters

|          |                                                                               |
|----------|-------------------------------------------------------------------------------|
| <i>x</i> | Number of pixels to move the window on the <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i> | Number of pixels to move the window on the <b>Y axis</b><br>Range: -128 - 127 |

## See also

[move\\_win](#)

**20.42.4.69 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 tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank
- `VBK_REG = VBK_BANK_1` indicates the second

**20.42.4.70 `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](#)

**20.42.4.71 `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                |
| <i>nb_tiles</i>   | Number of tiles to read                             |
| <i>data</i>       | Pointer to destination buffer for Tile Pattern data |

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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.

**20.42.4.72 `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

**20.42.4.73 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<br>In CGB Mode this could be either in VRAM Bank<br>0 or 1, depending on Bit 3 of the OAM Attribute Flag<br>(see <a href="#">set_sprite_prop</a> ) |

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](#)

**20.42.4.74 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

**20.42.4.75 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.

It's recommended to use GBDK constants (eg: `S_FLIPY`) to configure sprite properties as these are crossplatform.

```
// Load palette data into the first palette
set_sprite_palette(4, 1, exampleSprite_palettes)
// Set the OAM value for the sprite
// These flags tell the sprite to flip both vertically and horizontally.
set_sprite_prop(0, S_FLIPY | S_FLIPX);
```

See also

[S\\_PALETTE](#), [S\\_FLIPX](#), [S\\_FLIPY](#), [S\\_PRIORITY](#)

**20.42.4.76** `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

**20.42.4.77** `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).<br>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).<br>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.

**20.42.4.78 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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i>  | Number of pixels to move the sprite on the <b>Y axis</b><br>Range: -128 - 127 |

#### See also

[move\\_sprite](#) for more details about the X and Y position

**20.42.4.79 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](#)

**20.42.4.80 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 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](#)

**20.42.4.81 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        |

Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

Copies **len** bytes from VRAM starting at **vram\_addr** into a buffer at **data**.

GBC only: [VBK\\_REG](#) determines which bank of 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](#)

**20.42.4.82 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 tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank
- `VBK_REG = VBK_BANK_1` indicates the second

**20.42.4.83 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](#)

**20.42.4.84 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](#)

**20.42.4.85 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 |
| <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                   |

#### Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.42.4.86 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

**20.42.4.87 set\_bkg\_native\_data()** `void set_bkg_native_data (`  
`uint8_t first_tile,`  
`uint8_t nb_tiles,`  
`const uint8_t * data ) [inline]`

Sets VRAM Tile Pattern data for the Background / Window in the native format



## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.  
GBC only: [VBK\\_REG](#) determines which bank of 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](#)

**20.42.4.88 set\_sprite\_native\_data()** `void set_sprite_native_data (`  
    `uint8_t first_tile,`  
    `uint8_t nb_tiles,`  
    `const uint8_t * data ) [inline]`

Sets VRAM Tile Pattern data for Sprites in the native format

## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.  
GBC only: [VBK\\_REG](#) determines which bank of tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank
- `VBK_REG = VBK_BANK_1` indicates the second

**20.42.4.89 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

**20.42.4.90 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

**20.42.4.91 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

**20.42.4.92 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                                                        |

**20.42.4.93 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 |
|----------|---------------------------------------------------------------|

Parameters

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <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                                                    |

20.42.5 Variable Documentation

20.42.5.1 **c** void c

20.42.5.2 **d** void d

20.42.5.3 **e** void e

20.42.5.4 **h** void h

20.42.5.5 **l** void l

Initial value:

```
{
    __asm__("ei")
}
```

20.42.5.6 **\_cpu** uint8\_t \_cpu [extern]  
GB CPU type

See also

[DMG\\_TYPE](#), [MGB\\_TYPE](#), [CGB\\_TYPE](#), [cpu\\_fast\(\)](#), [cpu\\_slow\(\)](#), [\\_is\\_GBA](#)

20.42.5.7 **\_is\_GBA** uint8\_t \_is\_GBA [extern]  
GBA detection

See also

[GBA\\_DETECTED](#), [GBA\\_NOT\\_DETECTED](#), [\\_cpu](#)

20.42.5.8 **sys\_time** volatile uint16\_t sys\_time [extern]  
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)

20.42.5.9 **\_io\_status** volatile uint8\_t \_io\_status [extern]  
Serial Link: Current IO Status. An OR of IO\_\*

20.42.5.10 **\_io\_in** volatile uint8\_t \_io\_in [extern]  
Serial Link: Byte just read after calling [receive\\_byte\(\)](#)

**20.42.5.11** `_io_out` `volatile uint8_t _io_out` [extern]

Serial Link: Write byte to send here before calling `send_byte()`

**20.42.5.12** `_current_bank` `__REG _current_bank`

Tracks current active ROM bank

In most cases the `CURRENT_BANK` macro for this variable is recommended for use instead of the variable itself.

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()`

**20.42.5.13** `b` `void b`

**20.42.5.14** `_current_1bpp_colors` `uint16_t _current_1bpp_colors` [extern]

**20.42.5.15** `_map_tile_offset` `uint8_t _map_tile_offset` [extern]

**20.42.5.16** `_submap_tile_offset` `uint8_t _submap_tile_offset` [extern]

**20.42.5.17** `shadow_OAM` `volatile struct OAM_item_t shadow_OAM[]` [extern]

Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

**20.42.5.18** `_shadow_OAM_base` `__REG _shadow_OAM_base`

MSB of shadow\_OAM address is used by OAM DMA copying routine

## 20.43 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`)
- `void gb_decompress_bkg_data` (`uint8_t first_tile`, `const uint8_t *sour`)
- `void gb_decompress_win_data` (`uint8_t first_tile`, `const uint8_t *sour`)
- `void gb_decompress_sprite_data` (`uint8_t first_tile`, `const uint8_t *sour`)

### 20.43.1 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

### 20.43.2 Function Documentation

**20.43.2.1 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](#)

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](#)

**20.43.2.2 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](#)

**20.43.2.3 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](#)

**20.43.2.4 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](#)

**20.44 gbdk-lib/include/gbdk/gbdecompress.h File Reference**

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

**20.45 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](#)

**20.45.1 Function Documentation**

**20.45.1.1 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](#)

## 20.45.2 Variable Documentation

**20.45.2.1** `c uint16_t c`

## 20.46 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 SIOF_XFER_START 0b10000000`
- `#define SIOF_CLOCK_INT 0b00000001`
- `#define SIOF_CLOCK_EXT 0b00000000`
- `#define SIOF_SPEED_1X 0b00000000`
- `#define SIOF_SPEED_32X 0b00000010`
- `#define SIOF_B_CLOCK 0`
- `#define SIOF_B_SPEED 1`
- `#define SIOF_B_XFER_START 7`
- `#define SCF_START SIOF_XFER_START`
- `#define SCF_SOURCE SIOF_CLOCK_INT`

- #define SCF\_SPEED SIOF\_SPEED\_32X
- #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 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 rLCDC LCDC\_REG
- #define LCDCF\_OFF 0b00000000



- #define `LDCDF_ON` 0b10000000
- #define `LDCDF_WIN9800` 0b00000000
- #define `LDCDF_WIN9C00` 0b01000000
- #define `LDCDF_WINOFF` 0b00000000
- #define `LDCDF_WINON` 0b00100000
- #define `LDCDF_BG8800` 0b00000000
- #define `LDCDF_BG8000` 0b00010000
- #define `LDCDF_BG9800` 0b00000000
- #define `LDCDF_BG9C00` 0b00001000
- #define `LDCDF_OBJ8` 0b00000000
- #define `LDCDF_OBJ16` 0b00000100
- #define `LDCDF_OBJOFF` 0b00000000
- #define `LDCDF_OBJON` 0b00000010
- #define `LDCDF_BGOFF` 0b00000000
- #define `LDCDF_BGON` 0b00000001
- #define `LDCDF_B_ON` 7
- #define `LDCDF_B_WIN9C00` 6
- #define `LDCDF_B_WINON` 5
- #define `LDCDF_B_BG8000` 4
- #define `LDCDF_B_BG9C00` 3
- #define `LDCDF_B_OBJ16` 2
- #define `LDCDF_B_OBJON` 1
- #define `LDCDF_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_DBLSPD` 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\_AUD3WAVRAM []
- \_\_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](#)

### 20.46.1 Detailed Description

Defines that let the GB's hardware registers be accessed from C.  
See the [Pandocs](#) for more details on each register.

### 20.46.2 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** `__REG` `#define __REG extern volatile SFR`

**20.46.2.4** `rP1` `#define rP1 P1_REG`

**20.46.2.5** `P1F_5` `#define P1F_5 0b00100000`

**20.46.2.6** `P1F_4` `#define P1F_4 0b00010000`

**20.46.2.7** `P1F_3` `#define P1F_3 0b00001000`

**20.46.2.8** `P1F_2` `#define P1F_2 0b00000100`

**20.46.2.9** `P1F_1` `#define P1F_1 0b00000010`

**20.46.2.10** `P1F_0` `#define P1F_0 0b00000001`

**20.46.2.11** `P1F_GET_DPAD` `#define P1F_GET_DPAD P1F_5`

**20.46.2.12** `P1F_GET_BTN` `#define P1F_GET_BTN P1F_4`

**20.46.2.13** `P1F_GET_NONE` `#define P1F_GET_NONE (P1F_4 | P1F_5)`

**20.46.2.14** `rSB` `#define rSB SB_REG`

**20.46.2.15** `rSC` `#define rSC SC_REG`

**20.46.2.16 SIOF\_XFER\_START** #define SIOF\_XFER\_START 0b10000000  
Serial IO: Start Transfer. Automatically cleared at the end of transfer

**20.46.2.17 SIOF\_CLOCK\_INT** #define SIOF\_CLOCK\_INT 0b00000001  
Serial IO: Use Internal clock

**20.46.2.18 SIOF\_CLOCK\_EXT** #define SIOF\_CLOCK\_EXT 0b00000000  
Serial IO: Use External clock

**20.46.2.19 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)

**20.46.2.20 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)

**20.46.2.21 SIOF\_B\_CLOCK** #define SIOF\_B\_CLOCK 0

**20.46.2.22 SIOF\_B\_SPEED** #define SIOF\_B\_SPEED 1

**20.46.2.23 SIOF\_B\_XFER\_START** #define SIOF\_B\_XFER\_START 7

**20.46.2.24 SCF\_START** #define SCF\_START [SIOF\\_XFER\\_START](#)

**20.46.2.25 SCF\_SOURCE** #define SCF\_SOURCE [SIOF\\_CLOCK\\_INT](#)

**20.46.2.26 SCF\_SPEED** #define SCF\_SPEED [SIOF\\_SPEED\\_32X](#)

**20.46.2.27 rDIV** #define rDIV [DIV\\_REG](#)

**20.46.2.28 rTIMA** #define rTIMA [TIMA\\_REG](#)

**20.46.2.29 rTMA** #define rTMA [TMA\\_REG](#)

**20.46.2.30 rTAC** #define rTAC [TAC\\_REG](#)

**20.46.2.31 TACF\_START** #define TACF\_START 0b00000100

**20.46.2.32 TACF\_STOP** #define TACF\_STOP 0b00000000

**20.46.2.33 TACF\_4KHZ** #define TACF\_4KHZ 0b00000000

**20.46.2.34 TACF\_16KHZ** `#define TACF_16KHZ 0b00000011`

**20.46.2.35 TACF\_65KHZ** `#define TACF_65KHZ 0b00000010`

**20.46.2.36 TACF\_262KHZ** `#define TACF_262KHZ 0b00000001`

**20.46.2.37 rIF** `#define rIF IF_REG`

**20.46.2.38 rAUD1SWEEP** `#define rAUD1SWEEP NR10_REG`

Sound Channel 1, NR10: Sweep

**20.46.2.39 AUD1SWEEP\_UP** `#define AUD1SWEEP_UP 0b00000000`

For Sound Channel 1, NR10: Sweep Addition, period increases

**20.46.2.40 AUD1SWEEP\_DOWN** `#define AUD1SWEEP_DOWN 0b00001000`

For Sound Channel 1, NR10: Sweep Subtraction, period decreases

**20.46.2.41 AUD1SWEEP\_TIME** `#define AUD1SWEEP_TIME(`

`x ) ((x) << 4)`

For Sound Channel 1, NR10: Sweep Time/Pace, Range: 0-7

**20.46.2.42 AUD1SWEEP\_LENGTH** `#define AUD1SWEEP_LENGTH(`

`x ) (x)`

For Sound Channel 1, NR10: Sweep Length/Individual step, Range: 0-7

**20.46.2.43 rAUD1LEN** `#define rAUD1LEN NR11_REG`

Sound Channel 1, NR11: Sound length/Wave pattern duty

**20.46.2.44 rAUD1ENV** `#define rAUD1ENV NR12_REG`

Sound Channel 1, NR12: Volume Envelope

**20.46.2.45 rAUD1LOW** `#define rAUD1LOW NR13_REG`

Sound Channel 1, NR13: Frequency Low

**20.46.2.46 rAUD1HIGH** `#define rAUD1HIGH NR14_REG`

Sound Channel 1, NR14: Frequency High

**20.46.2.47 rAUD2LEN** `#define rAUD2LEN NR21_REG`

Sound Channel 2, NR21\_REG: Tone

**20.46.2.48 rAUD2ENV** `#define rAUD2ENV NR22_REG`

Sound Channel 2, NR22\_REG: Volume Envelope

**20.46.2.49 rAUD2LOW** `#define rAUD2LOW NR23_REG`

Sound Channel 2, NR23\_REG: Frequency data Low

**20.46.2.50 rAUD2HIGH** `#define rAUD2HIGH NR24_REG`

Sound Channel 2, NR24\_REG: Frequency data High

**20.46.2.51 rAUD3ENA** `#define rAUD3ENA NR30_REG`  
Sound Channel 3, NR30\_REG: Sound on/off

**20.46.2.52 rAUD3LEN** `#define rAUD3LEN NR31_REG`  
Sound Channel 3, NR31\_REG: Sound Length

**20.46.2.53 rAUD3LEVEL** `#define rAUD3LEVEL NR32_REG`  
Sound Channel 3, NR32\_REG: Select output level

**20.46.2.54 rAUD3LOW** `#define rAUD3LOW NR33_REG`  
Sound Channel 3, NR33\_REG: Frequency data Low

**20.46.2.55 rAUD3HIGH** `#define rAUD3HIGH NR34_REG`  
Sound Channel 3, NR34\_REG: Frequency data High

**20.46.2.56 rAUD4LEN** `#define rAUD4LEN NR41_REG`  
Sound Channel 4, NR41\_REG: Sound Length

**20.46.2.57 rAUD4ENV** `#define rAUD4ENV NR42_REG`  
Sound Channel 4, NR42\_REG: Volume Envelope

**20.46.2.58 rAUD4POLY** `#define rAUD4POLY NR43_REG`  
Sound Channel 4, NR43\_REG: Polynomial Counter

**20.46.2.59 AUD4POLY\_WIDTH\_15BIT** `#define AUD4POLY_WIDTH_15BIT 0x00`  
For Sound Channel 4, NR43\_REG: Polynomial counter use 15 steps

**20.46.2.60 AUD4POLY\_WIDTH\_7BIT** `#define AUD4POLY_WIDTH_7BIT 0x08`  
For Sound Channel 4, NR43\_REG: Polynomial counter use 7 steps

**20.46.2.61 rAUD4GO** `#define rAUD4GO NR44_REG`  
Sound Channel 4, NR44\_REG: Counter / Consecutive and Initial

**20.46.2.62 rAUDVOL** `#define rAUDVOL NR50_REG`  
Sound Master Volume, NR50: Volume and Cart external sound input (VIN)

**20.46.2.63 AUDVOL\_VOL\_LEFT** `#define AUDVOL_VOL_LEFT(`  
                  `x ) ((x) << 4)`  
For Sound Master Volume, NR50: Left Volume, Range: 0-7

**20.46.2.64 AUDVOL\_VOL\_RIGHT** `#define AUDVOL_VOL_RIGHT(`  
                  `x ) ((x))`  
For Sound Master Volume, NR50: Right Volume, Range: 0-7

**20.46.2.65 AUDVOL\_VIN\_LEFT** `#define AUDVOL_VIN_LEFT 0b10000000`  
For Sound Master Volume, NR50: Cart external sound input (VIN) Left bit, 1 = ON, 0 = OFF

**20.46.2.66 AUDVOL\_VIN\_RIGHT** `#define AUDVOL_VIN_RIGHT 0b00001000`  
For Sound Master Volume, NR50: Cart external sound input (VIN) Right bit, 1 = ON, 0 = OFF



**20.46.2.67 rAUDTERM** `#define rAUDTERM NR51_REG`

Sound Panning, NR51: Enable/disable left and right output for sound channels

**20.46.2.68 AUDTERM\_4\_LEFT** `#define AUDTERM_4_LEFT 0b10000000`

For Sound Panning, NR51: Channel 4 Left bit, 1 = ON, 0 = OFF

**20.46.2.69 AUDTERM\_3\_LEFT** `#define AUDTERM_3_LEFT 0b01000000`

For Sound Panning, NR51: Channel 3 Left bit, 1 = ON, 0 = OFF

**20.46.2.70 AUDTERM\_2\_LEFT** `#define AUDTERM_2_LEFT 0b00100000`

For Sound Panning, NR51: Channel 2 Left bit, 1 = ON, 0 = OFF

**20.46.2.71 AUDTERM\_1\_LEFT** `#define AUDTERM_1_LEFT 0b00010000`

For Sound Panning, NR51: Channel 1 Left bit, 1 = ON, 0 = OFF

**20.46.2.72 AUDTERM\_4\_RIGHT** `#define AUDTERM_4_RIGHT 0b00001000`

For Sound Panning, NR51: Channel 4 Right bit, 1 = ON, 0 = OFF

**20.46.2.73 AUDTERM\_3\_RIGHT** `#define AUDTERM_3_RIGHT 0b00000100`

For Sound Panning, NR51: Channel 3 Right bit, 1 = ON, 0 = OFF

**20.46.2.74 AUDTERM\_2\_RIGHT** `#define AUDTERM_2_RIGHT 0b00000010`

For Sound Panning, NR51: Channel 2 Right bit, 1 = ON, 0 = OFF

**20.46.2.75 AUDTERM\_1\_RIGHT** `#define AUDTERM_1_RIGHT 0b00000001`

For Sound Panning, NR51: Channel 1 Right bit, 1 = ON, 0 = OFF

**20.46.2.76 rAUDENA** `#define rAUDENA NR52_REG`

Sound Master Control, NR52: ON / OFF

**20.46.2.77 AUDENA\_ON** `#define AUDENA_ON 0b10000000`

For Sound Master Control, NR52: Sound ON

**20.46.2.78 AUDENA\_OFF** `#define AUDENA_OFF 0b00000000`

For Sound Master Control, NR52: Sound OFF

**20.46.2.79 rLCDC** `#define rLCDC LCDC_REG`**20.46.2.80 LCDCF\_OFF** `#define LCDCF_OFF 0b00000000`

LCD Control: Off

**20.46.2.81 LCDCF\_ON** `#define LCDCF_ON 0b10000000`

LCD Control: On

**20.46.2.82 LCDCF\_WIN9800** `#define LCDCF_WIN9800 0b00000000`

Window Tile Map: Use 9800 Region

**20.46.2.83 LCDCF\_WIN9C00** `#define LCDCF_WIN9C00 0b01000000`

Window Tile Map: Use 9C00 Region

**20.46.2.84 LCDCF\_WINOFF** `#define LCDCF_WINOFF 0b00000000`

Window Display: Hidden

**20.46.2.85 LCDCF\_WINON** `#define LCDCF_WINON 0b00100000`  
Window Display: Visible

**20.46.2.86 LCDCF\_BG8800** `#define LCDCF_BG8800 0b00000000`  
BG & Window Tile Data: Use 8800 Region

**20.46.2.87 LCDCF\_BG8000** `#define LCDCF_BG8000 0b00010000`  
BG & Window Tile Data: Use 8000 Region

**20.46.2.88 LCDCF\_BG9800** `#define LCDCF_BG9800 0b00000000`  
BG Tile Map: use 9800 Region

**20.46.2.89 LCDCF\_BG9C00** `#define LCDCF_BG9C00 0b00001000`  
BG Tile Map: use 9C00 Region

**20.46.2.90 LCDCF\_OBJ8** `#define LCDCF_OBJ8 0b00000000`  
Sprites Size: 8x8 pixels

**20.46.2.91 LCDCF\_OBJ16** `#define LCDCF_OBJ16 0b00000100`  
Sprites Size: 8x16 pixels

**20.46.2.92 LCDCF\_OBJOFF** `#define LCDCF_OBJOFF 0b00000000`  
Sprites Display: Hidden

**20.46.2.93 LCDCF\_OBJON** `#define LCDCF_OBJON 0b00000010`  
Sprites Display: Visible

**20.46.2.94 LCDCF\_BG0FF** `#define LCDCF_BG0FF 0b00000000`  
Background Display: Hidden

**20.46.2.95 LCDCF\_BGON** `#define LCDCF_BGON 0b00000001`  
Background Display: Visible

**20.46.2.96 LCDCF\_B\_ON** `#define LCDCF_B_ON 7`  
Bit for LCD On/Off Select

**20.46.2.97 LCDCF\_B\_WIN9C00** `#define LCDCF_B_WIN9C00 6`  
Bit for Window Tile Map Region Select

**20.46.2.98 LCDCF\_B\_WINON** `#define LCDCF_B_WINON 5`  
Bit for Window Display On/Off Control

**20.46.2.99 LCDCF\_B\_BG8000** `#define LCDCF_B_BG8000 4`  
Bit for BG & Window Tile Data Region Select

**20.46.2.100 LCDCF\_B\_BG9C00** `#define LCDCF_B_BG9C00 3`  
Bit for BG Tile Map Region Select

**20.46.2.101 LCDCF\_B\_OBJ16** `#define LCDCF_B_OBJ16 2`  
Bit for Sprites Size Select

**20.46.2.102 LCDCF\_B\_OBJON** `#define LCDCF_B_OBJON 1`  
Bit for Sprites Display Visible/Hidden Select

**20.46.2.103 LCDCF\_B\_BGON** `#define LCDCF_B_BGON 0`  
Bit for Background Display Visible/Hidden Select

**20.46.2.104 rSTAT** `#define rSTAT STAT_REG`

**20.46.2.105 STATF\_LYC** `#define STATF_LYC 0b01000000`  
STAT Interrupt: LYC=LY Coincidence Source Enable

**20.46.2.106 STATF\_MODE10** `#define STATF_MODE10 0b00100000`  
STAT Interrupt: Mode 2 OAM Source Enable

**20.46.2.107 STATF\_MODE01** `#define STATF_MODE01 0b00010000`  
STAT Interrupt: Mode 1 VBlank Source Enable

**20.46.2.108 STATF\_MODE00** `#define STATF_MODE00 0b00001000`  
STAT Interrupt: Mode 0 HBlank Source Enable

**20.46.2.109 STATF\_LYCF** `#define STATF_LYCF 0b00000100`  
LYC=LY Coincidence Status Flag, Set when LY contains the same value as LYC

**20.46.2.110 STATF\_HBL** `#define STATF_HBL 0b00000000`  
Current LCD Mode is: 0, in H-Blank

**20.46.2.111 STATF\_VBL** `#define STATF_VBL 0b00000001`  
Current LCD Mode is: 1, in V-Blank

**20.46.2.112 STATF\_OAM** `#define STATF_OAM 0b00000010`  
Current LCD Mode is: 2, in OAM-RAM is used by system (Searching OAM)

**20.46.2.113 STATF\_LCD** `#define STATF_LCD 0b00000011`  
Current LCD Mode is: 3, both OAM and VRAM used by system (Transferring Data to LCD Controller)

**20.46.2.114 STATF\_BUSY** `#define STATF_BUSY 0b00000010`  
When set, VRAM access is unsafe

**20.46.2.115 STATF\_B\_LYC** `#define STATF_B_LYC 6`  
Bit for STAT Interrupt: LYC=LY Coincidence Source Enable

**20.46.2.116 STATF\_B\_MODE10** `#define STATF_B_MODE10 5`  
Bit for STAT Interrupt: Mode 2 OAM Source Enable

**20.46.2.117 STATF\_B\_MODE01** `#define STATF_B_MODE01 4`  
Bit for STAT Interrupt: Mode 1 VBlank Source Enable

**20.46.2.118 STATF\_B\_MODE00** `#define STATF_B_MODE00 3`  
Bit for STAT Interrupt: Mode 0 HBlank Source Enable

**20.46.2.119 STATF\_B\_LYCF** `#define STATF_B_LYCF 2`  
Bit for LYC=LY Coincidence Status Flag

**20.46.2.120 STATF\_B\_VBL** `#define STATF_B_VBL 0`

**20.46.2.121 STATF\_B\_OAM** `#define STATF_B_OAM 1`

**20.46.2.122 STATF\_B\_BUSY** `#define STATF_B_BUSY 1`  
Bit for when VRAM access is unsafe

**20.46.2.123 rSCY** `#define rSCY`

**20.46.2.124 rSCX** `#define rSCX SCX_REG`

**20.46.2.125 rLY** `#define rLY LY_REG`

**20.46.2.126 rLYC** `#define rLYC LYC_REG`

**20.46.2.127 rDMA** `#define rDMA DMA_REG`

**20.46.2.128 rBGP** `#define rBGP BGP_REG`

**20.46.2.129 rOBP0** `#define rOBP0 OBP0_REG`

**20.46.2.130 rOBP1** `#define rOBP1 OBP1_REG`

**20.46.2.131 rWY** `#define rWY WY_REG`

**20.46.2.132 rWX** `#define rWX WX_REG`

**20.46.2.133 rKEY1** `#define rKEY1 KEY1_REG`

**20.46.2.134 rSPD** `#define rSPD KEY1_REG`

**20.46.2.135 KEY1F\_DBLSPPEED** `#define KEY1F_DBLSPPEED 0b10000000`

**20.46.2.136 KEY1F\_PREPARE** `#define KEY1F_PREPARE 0b00000001`

**20.46.2.137 rVBK** `#define rVBK VBK_REG`

**20.46.2.138 VBK\_BANK\_0** `#define VBK_BANK_0 0`  
Select Regular Map and Normal Tiles (CGB Mode Only)

**20.46.2.139 VBK\_TILES** `#define VBK_TILES 0`  
Select Regular Map and Normal Tiles (CGB Mode Only)

**20.46.2.140 VBK\_BANK\_1** `#define VBK_BANK_1 1`  
Select Map Attributes and Extra Tile Bank (CGB Mode Only)

**20.46.2.141 VBK\_ATTRIBUTES** `#define VBK_ATTRIBUTES 1`  
Select Map Attributes and Extra Tile Bank (CGB Mode Only)

**20.46.2.142 BKGF\_PRI** `#define BKGF_PRI 0b10000000`  
Background CGB BG and Window over Sprite priority Enabled

**20.46.2.143 BKGF\_YFLIP** `#define BKGF_YFLIP 0b01000000`  
Background CGB Y axis flip: Vertically mirrored

**20.46.2.144 BKGF\_XFLIP** `#define BKGF_XFLIP 0b00100000`  
Background CGB X axis flip: Horizontally mirrored

**20.46.2.145 BKGF\_BANK0** `#define BKGF_BANK0 0b00000000`  
Background CGB Tile VRAM-Bank: Use Bank 0 (CGB Mode Only)

**20.46.2.146 BKGF\_BANK1** `#define BKGF_BANK1 0b00001000`  
Background CGB Tile VRAM-Bank: Use Bank 1 (CGB Mode Only)

**20.46.2.147 BKGF\_CGB\_PAL0** `#define BKGF_CGB_PAL0 0b00000000`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.148 BKGF\_CGB\_PAL1** `#define BKGF_CGB_PAL1 0b00000001`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.149 BKGF\_CGB\_PAL2** `#define BKGF_CGB_PAL2 0b00000010`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.150 BKGF\_CGB\_PAL3** `#define BKGF_CGB_PAL3 0b00000011`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.151 BKGF\_CGB\_PAL4** `#define BKGF_CGB_PAL4 0b00000100`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.152 BKGF\_CGB\_PAL5** `#define BKGF_CGB_PAL5 0b00000101`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.153 BKGF\_CGB\_PAL6** `#define BKGF_CGB_PAL6 0b00000110`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.154 BKGF\_CGB\_PAL7** `#define BKGF_CGB_PAL7 0b00000111`  
Background CGB Palette number (CGB Mode Only)

**20.46.2.155 rHDMAl** `#define rHDMAl HDMAl\_REG`

**20.46.2.156 rDMA2** #define rDMA2 [DMA2\\_REG](#)

**20.46.2.157 rDMA3** #define rDMA3 [DMA3\\_REG](#)

**20.46.2.158 rDMA4** #define rDMA4 [DMA4\\_REG](#)

**20.46.2.159 rDMA5** #define rDMA5 [DMA5\\_REG](#)

**20.46.2.160 DMA5F\_MODE\_GP** #define DMA5F\_MODE\_GP 0b00000000

**20.46.2.161 DMA5F\_MODE\_HBL** #define DMA5F\_MODE\_HBL 0b10000000

**20.46.2.162 DMA5F\_BUSY** #define DMA5F\_BUSY 0b10000000

**20.46.2.163 rRP** #define rRP [RP\\_REG](#)

**20.46.2.164 RPF\_ENREAD** #define RPF\_ENREAD 0b11000000

**20.46.2.165 RPF\_DATAIN** #define RPF\_DATAIN 0b00000010

**20.46.2.166 RPF\_WRITE\_HI** #define RPF\_WRITE\_HI 0b00000001

**20.46.2.167 RPF\_WRITE\_LO** #define RPF\_WRITE\_LO 0b00000000

**20.46.2.168 rBCPS** #define rBCPS [BCPS\\_REG](#)

**20.46.2.169 BCPSF\_AUTOINC** #define BCPSF\_AUTOINC 0b10000000

**20.46.2.170 rBCPD** #define rBCPD [BCPD\\_REG](#)

**20.46.2.171 rOCPS** #define rOCPS [OCPS\\_REG](#)

**20.46.2.172 OCPSF\_AUTOINC** #define OCPSF\_AUTOINC 0b10000000

**20.46.2.173 rOCPD** #define rOCPD [OCPD\\_REG](#)

**20.46.2.174 rSVBK** `#define rSVBK SVBK_REG`

**20.46.2.175 rSMBK** `#define rSMBK SVBK_REG`

**20.46.2.176 rPCM12** `#define rPCM12 PCM12_REG`

**20.46.2.177 rPCM34** `#define rPCM34 PCM34_REG`

**20.46.2.178 rIE** `#define rIE IE_REG`

**20.46.2.179 IEF\_HILO** `#define IEF_HILO 0b00010000`  
Joypad interrupt enable flag

**20.46.2.180 IEF\_SERIAL** `#define IEF_SERIAL 0b00001000`  
Serial interrupt enable flag

**20.46.2.181 IEF\_TIMER** `#define IEF_TIMER 0b00000100`  
Timer interrupt enable flag

**20.46.2.182 IEF\_STAT** `#define IEF_STAT 0b00000010`  
Stat interrupt enable flag

**20.46.2.183 IEF\_VBLANK** `#define IEF_VBLANK 0b00000001`  
VBlank interrupt enable flag

**20.46.2.184 AUDLEN\_DUTY\_12\_5** `#define AUDLEN_DUTY_12_5 0b00000000`

**20.46.2.185 AUDLEN\_DUTY\_25** `#define AUDLEN_DUTY_25 0b01000000`

**20.46.2.186 AUDLEN\_DUTY\_50** `#define AUDLEN_DUTY_50 0b10000000`

**20.46.2.187 AUDLEN\_DUTY\_75** `#define AUDLEN_DUTY_75 0b11000000`

**20.46.2.188 AUDLEN\_LENGTH** `#define AUDLEN_LENGTH(  
x ) (x)`

**20.46.2.189 AUDENV\_VOL** `#define AUDENV_VOL(  
x ) ((x) << 4)`

**20.46.2.190 AUDENV\_UP** `#define AUDENV_UP 0b00001000`

**20.46.2.191 AUDENV\_DOWN** #define AUDENV\_DOWN 0b00000000

**20.46.2.192 AUDENV\_LENGTH** #define AUDENV\_LENGTH(  
x ) (x)

**20.46.2.193 AUDHIGH\_RESTART** #define AUDHIGH\_RESTART 0b10000000

**20.46.2.194 AUDHIGH\_LENGTH\_ON** #define AUDHIGH\_LENGTH\_ON 0b01000000

**20.46.2.195 AUDHIGH\_LENGTH\_OFF** #define AUDHIGH\_LENGTH\_OFF 0b00000000

**20.46.2.196 OAMF\_PRI** #define OAMF\_PRI 0b10000000  
BG and Window over Sprite Enabled

**20.46.2.197 OAMF\_YFLIP** #define OAMF\_YFLIP 0b01000000  
Sprite Y axis flip: Vertically mirrored

**20.46.2.198 OAMF\_XFLIP** #define OAMF\_XFLIP 0b00100000  
Sprite X axis flip: Horizontally mirrored

**20.46.2.199 OAMF\_PAL0** #define OAMF\_PAL0 0b00000000  
Sprite Palette number: use OBP0 (Non-CGB Mode Only)

**20.46.2.200 OAMF\_PAL1** #define OAMF\_PAL1 0b00010000  
Sprite Palette number: use OBP1 (Non-CGB Mode Only)

**20.46.2.201 OAMF\_BANK0** #define OAMF\_BANK0 0b00000000  
Sprite Tile VRAM-Bank: Use Bank 0 (CGB Mode Only)

**20.46.2.202 OAMF\_BANK1** #define OAMF\_BANK1 0b00001000  
Sprite Tile VRAM-Bank: Use Bank 1 (CGB Mode Only)

**20.46.2.203 OAMF\_CGB\_PAL0** #define OAMF\_CGB\_PAL0 0b00000000  
Sprite CGB Palette number: use OCP0 (CGB Mode Only)

**20.46.2.204 OAMF\_CGB\_PAL1** #define OAMF\_CGB\_PAL1 0b00000001  
Sprite CGB Palette number: use OCP1 (CGB Mode Only)

**20.46.2.205 OAMF\_CGB\_PAL2** #define OAMF\_CGB\_PAL2 0b00000010  
Sprite CGB Palette number: use OCP2 (CGB Mode Only)

**20.46.2.206 OAMF\_CGB\_PAL3** #define OAMF\_CGB\_PAL3 0b00000011  
Sprite CGB Palette number: use OCP3 (CGB Mode Only)

**20.46.2.207 OAMF\_CGB\_PAL4** #define OAMF\_CGB\_PAL4 0b00000100  
Sprite CGB Palette number: use OCP4 (CGB Mode Only)

**20.46.2.208 OAMF\_CGB\_PAL5** #define OAMF\_CGB\_PAL5 0b00000101  
Sprite CGB Palette number: use OCP5 (CGB Mode Only)



**20.46.2.209 OAMF\_CGB\_PAL6** `#define OAMF_CGB_PAL6 0b00000110`  
Sprite CGB Palette number: use OCP6 (CGB Mode Only)

**20.46.2.210 OAMF\_CGB\_PAL7** `#define OAMF_CGB_PAL7 0b00000111`  
Sprite CGB Palette number: use OCP7 (CGB Mode Only)

**20.46.2.211 OAMF\_PALMASK** `#define OAMF_PALMASK 0b00000111`  
Mask for Sprite CGB Palette number (CGB Mode Only)

**20.46.2.212 DEVICE\_SCREEN\_X\_OFFSET** `#define DEVICE_SCREEN_X_OFFSET 0`  
Offset of visible screen (in tile units) from left edge of hardware map

**20.46.2.213 DEVICE\_SCREEN\_Y\_OFFSET** `#define DEVICE_SCREEN_Y_OFFSET 0`  
Offset of visible screen (in tile units) from top edge of hardware map

**20.46.2.214 DEVICE\_SCREEN\_WIDTH** `#define DEVICE_SCREEN_WIDTH 20`  
Width of visible screen in tile units

**20.46.2.215 DEVICE\_SCREEN\_HEIGHT** `#define DEVICE_SCREEN_HEIGHT 18`  
Height of visible screen in tile units

**20.46.2.216 DEVICE\_SCREEN\_BUFFER\_WIDTH** `#define DEVICE_SCREEN_BUFFER_WIDTH 32`  
Width of hardware map buffer in tile units

**20.46.2.217 DEVICE\_SCREEN\_BUFFER\_HEIGHT** `#define DEVICE_SCREEN_BUFFER_HEIGHT 32`  
Height of hardware map buffer in tile units

**20.46.2.218 DEVICE\_SCREEN\_MAP\_ENTRY\_SIZE** `#define DEVICE_SCREEN_MAP_ENTRY_SIZE 1`  
Number of bytes per hardware map entry

**20.46.2.219 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

**20.46.2.220 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

**20.46.2.221 DEVICE\_WINDOW\_PX\_OFFSET\_X** `#define DEVICE_WINDOW_PX_OFFSET_X 7`  
Minimal X coordinate of the window layer

**20.46.2.222 DEVICE\_WINDOW\_PX\_OFFSET\_Y** `#define DEVICE_WINDOW_PX_OFFSET_Y 0`  
Minimal Y coordinate of the window layer

**20.46.2.223 DEVICE\_SCREEN\_PX\_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`  
Width of visible screen in pixels

**20.46.2.224 DEVICE\_SCREEN\_PX\_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`  
Height of visible screen in pixels

## 20.46.3 Variable Documentation

**20.46.3.1** `_VRAM` `__BYTES` `_VRAM[ ]`  
Memory map

**20.46.3.2** `_VRAM8000` `__BYTES` `_VRAM8000[ ]`

**20.46.3.3** `_VRAM8800` `__BYTES` `_VRAM8800[ ]`

**20.46.3.4** `_VRAM9000` `__BYTES` `_VRAM9000[ ]`

**20.46.3.5** `_SCRN0` `__BYTES` `_SCRN0[ ]`

**20.46.3.6** `_SCRN1` `__BYTES` `_SCRN1[ ]`

**20.46.3.7** `_SRAM` `__BYTES` `_SRAM[ ]`

**20.46.3.8** `_RAM` `__BYTES` `_RAM[ ]`

**20.46.3.9** `_RAMBANK` `__BYTES` `_RAMBANK[ ]`

**20.46.3.10** `_OAMRAM` `__BYTES` `_OAMRAM[ ]`

**20.46.3.11** `_IO` `__BYTE_REG` `_IO[ ]`

**20.46.3.12** `_AUD3WAVRAM` `__BYTE_REG` `_AUD3WAVRAM[ ]`

**20.46.3.13** `_HRAM` `__BYTE_REG` `_HRAM[ ]`

**20.46.3.14** `rRAMG` `__BYTE_REG` `rRAMG`  
MBC5 registers

**20.46.3.15** `rROMB0` `__BYTE_REG` `rROMB0`

**20.46.3.16** `rROMB1` `__BYTE_REG` `rROMB1`

**20.46.3.17** `rRAMB` `__BYTE_REG` `rRAMB`

**20.46.3.18 P1\_REG** [\\_\\_REG](#) P1\_REG

IO Registers Joystick register

See also

[joypad\(\)](#), [add\\_JOY\(\)](#), [IEF\\_HILO](#), [P1F\\_5](#), [P1F\\_4](#), [P1F\\_3](#), [P1F\\_2](#), [P1F\\_1](#), [P1F\\_0](#), [P1F\\_GET\\_DPAD](#), [P1F\\_GET\\_BTN](#), [P1F\\_GET\\_NONE](#)

**20.46.3.19 SB\_REG** [\\_\\_REG](#) SB\_REG

Serial IO data buffer

**20.46.3.20 SC\_REG** [\\_\\_REG](#) SC\_REG

Serial IO control register

**20.46.3.21 DIV\_REG** [\\_\\_REG](#) DIV\_REG

Divider register

**20.46.3.22 TIMA\_REG** [\\_\\_REG](#) TIMA\_REG

Timer counter

**20.46.3.23 TMA\_REG** [\\_\\_REG](#) TMA\_REG

Timer modulo

**20.46.3.24 TAC\_REG** [\\_\\_REG](#) TAC\_REG

Timer control

**20.46.3.25 IF\_REG** [\\_\\_REG](#) IF\_REG

Interrupt flags: [IEF\\_HILO](#), [IEF\\_SERIAL](#), [IEF\\_TIMER](#), [IEF\\_STAT](#), [IEF\\_VBLANK](#)

**20.46.3.26 NR10\_REG** [\\_\\_REG](#) NR10\_REG

Sound Channel 1, NR10: Sweep

**20.46.3.27 NR11\_REG** [\\_\\_REG](#) NR11\_REG

Sound Channel 1, NR11: Sound length/Wave pattern duty

**20.46.3.28 NR12\_REG** [\\_\\_REG](#) NR12\_REG

Sound Channel 1, NR12: Volume Envelope

**20.46.3.29 NR13\_REG** [\\_\\_REG](#) NR13\_REG

Sound Channel 1, NR13: Frequency Low

**20.46.3.30 NR14\_REG** [\\_\\_REG](#) NR14\_REG

Sound Channel 1, NR14: Frequency High

**20.46.3.31 NR21\_REG** [\\_\\_REG](#) NR21\_REG

Sound Channel 2, NR21\_REG: Tone

**20.46.3.32 NR22\_REG** [\\_\\_REG](#) NR22\_REG

Sound Channel 2, NR22\_REG: Volume Envelope

**20.46.3.33 NR23\_REG** [\\_\\_REG](#) NR23\_REG

Sound Channel 2, NR23\_REG: Frequency data Low

**20.46.3.34 NR24\_REG** [\\_\\_REG](#) NR24\_REG

Sound Channel 2, NR24\_REG: Frequency data High

**20.46.3.35 NR30\_REG** [\\_\\_REG](#) NR30\_REG

Sound Channel 3, NR30\_REG: Sound on/off

**20.46.3.36 NR31\_REG** [\\_\\_REG](#) NR31\_REG

Sound Channel 3, NR31\_REG: Sound Length

**20.46.3.37 NR32\_REG** [\\_\\_REG](#) NR32\_REG

Sound Channel 3, NR32\_REG: Select output level

**20.46.3.38 NR33\_REG** [\\_\\_REG](#) NR33\_REG

Sound Channel 3, NR33\_REG: Frequency data Low

**20.46.3.39 NR34\_REG** [\\_\\_REG](#) NR34\_REG

Sound Channel 3, NR34\_REG: Frequency data High

**20.46.3.40 NR41\_REG** [\\_\\_REG](#) NR41\_REG

Sound Channel 4, NR41\_REG: Sound Length

**20.46.3.41 NR42\_REG** [\\_\\_REG](#) NR42\_REG

Sound Channel 4, NR42\_REG: Volume Envelope

**20.46.3.42 NR43\_REG** [\\_\\_REG](#) NR43\_REG

Sound Channel 4, NR43\_REG: Polynomial Counter

**20.46.3.43 NR44\_REG** [\\_\\_REG](#) NR44\_REG

Sound Channel 4, NR44\_REG: Counter / Consecutive and Initial

**20.46.3.44 NR50\_REG** [\\_\\_REG](#) NR50\_REG

Sound Master Volume, NR50: Volume and Cart external sound input (VIN)

**20.46.3.45 NR51\_REG** [\\_\\_REG](#) NR51\_REG

Sound Panning, NR51: Enable/disable left and right output for sound channels

**20.46.3.46 NR52\_REG** [\\_\\_REG](#) NR52\_REG

Sound Master Control, NR52: ON / OFF

**20.46.3.47 AUD3WAVE** [\\_\\_BYTE\\_REG](#) AUD3WAVE[16]**20.46.3.48 PCM\_SAMPLE** [\\_\\_BYTE\\_REG](#) PCM\_SAMPLE[16]**20.46.3.49 LCDC\_REG** [\\_\\_REG](#) LCDC\_REG

LCD control

**20.46.3.50 STAT\_REG** [\\_\\_REG](#) STAT\_REG

LCD status

**20.46.3.51 SCY\_REG** [\\_\\_REG](#) SCY\_REG

Scroll Y

**20.46.3.52 SCX\_REG** [\\_\\_REG](#) SCX\_REG  
Scroll X

**20.46.3.53 LY\_REG** [\\_\\_REG](#) LY\_REG  
LCDC Y-coordinate

**20.46.3.54 LYC\_REG** [\\_\\_REG](#) LYC\_REG  
LY compare

**20.46.3.55 DMA\_REG** [\\_\\_REG](#) DMA\_REG  
DMA transfer

**20.46.3.56 BGP\_REG** [\\_\\_REG](#) BGP\_REG  
Set and Read the Background palette.

Example with the [DMG\\_PALETTE\(\)](#) helper function and constants:

```
BGP_REG = DMG_PALETTE(DMG_BLACK, DMG_DARK_GRAY, DMG_LITE_GRAY, DMG_WHITE);
```

**20.46.3.57 OBP0\_REG** [\\_\\_REG](#) OBP0\_REG  
Set and Read the OBJ (Sprite) palette 0.

The first color entry is always transparent.

Example with the [DMG\\_PALETTE\(\)](#) helper function and constants:

```
OBP0_REG = DMG_PALETTE(DMG_BLACK, DMG_DARK_GRAY, DMG_LITE_GRAY, DMG_WHITE);
```

**20.46.3.58 OBP1\_REG** [\\_\\_REG](#) OBP1\_REG  
Set and Read the OBJ (Sprite) palette 1.

The first color entry is always transparent.

Example with the [DMG\\_PALETTE\(\)](#) helper function and constants:

```
OBP1_REG = DMG_PALETTE(DMG_BLACK, DMG_DARK_GRAY, DMG_LITE_GRAY, DMG_WHITE);
```

**20.46.3.59 WY\_REG** [\\_\\_REG](#) WY\_REG  
Window Y coordinate

**20.46.3.60 WX\_REG** [\\_\\_REG](#) WX\_REG  
Window X coordinate

**20.46.3.61 KEY1\_REG** [\\_\\_REG](#) KEY1\_REG  
CPU speed

**20.46.3.62 VBK\_REG** [\\_\\_REG](#) VBK\_REG  
VRAM bank select (CGB only)

See also

[VBK\\_BANK\\_0](#), [VBK\\_TILES](#), [VBK\\_BANK\\_1](#), [VBK\\_ATTRIBUTES](#)

**20.46.3.63 HDMA1\_REG** [\\_\\_REG](#) HDMA1\_REG  
DMA control 1

**20.46.3.64 HDMA2\_REG** [\\_\\_REG](#) HDMA2\_REG  
DMA control 2

**20.46.3.65 HDMA3\_REG** [\\_\\_REG](#) HDMA3\_REG  
DMA control 3

**20.46.3.66 HDMA4\_REG** [\\_\\_REG](#) HDMA4\_REG  
DMA control 4

**20.46.3.67 HDMA5\_REG** [\\_\\_REG](#) HDMA5\_REG  
DMA control 5

**20.46.3.68 RP\_REG** [\\_\\_REG](#) RP\_REG  
IR port

**20.46.3.69 BCPS\_REG** [\\_\\_REG](#) BCPS\_REG  
BG color palette specification

**20.46.3.70 BCPD\_REG** [\\_\\_REG](#) BCPD\_REG  
BG color palette data

**20.46.3.71 OCPS\_REG** [\\_\\_REG](#) OCPS\_REG  
OBJ color palette specification

**20.46.3.72 OCPD\_REG** [\\_\\_REG](#) OCPD\_REG  
OBJ color palette data

**20.46.3.73 SVBK\_REG** [\\_\\_REG](#) SVBK\_REG  
Selects the WRAM upper region bank (CGB Only). WRAM Banking is NOT officially supported in GBDK and SDCC.  
The stack must be moved and other special care taken.

**20.46.3.74 PCM12\_REG** [\\_\\_REG](#) PCM12\_REG  
Sound channel 1&2 PCM amplitude (R)

**20.46.3.75 PCM34\_REG** [\\_\\_REG](#) PCM34\_REG  
Sound channel 3&4 PCM amplitude (R)

**20.46.3.76 IE\_REG** [\\_\\_REG](#) IE\_REG  
Interrupt enable

## 20.47 gbdk-lib/include/msx/hardware.h File Reference

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

### Macros

- `#define` [\\_\\_BYTES](#) extern [UBYTE](#)
- `#define` [\\_\\_BYTE\\_REG](#) extern volatile [UBYTE](#)
- `#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 0b00000010`
- `#define R0_SCR_MODE3 0b00000100`
- `#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 0b00000010`
- `#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 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 _SYSTEM`
- `volatile UBYTE VDP_ATTR_SHIFT`

### 20.47.1 Detailed Description

Defines that let the MSX 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 `PSG_LATCH`** `#define PSG_LATCH 0x80`

**20.47.2.4 `PSG_CH0`** `#define PSG_CH0 0b00000000`

**20.47.2.5 `PSG_CH1`** `#define PSG_CH1 0b00100000`

**20.47.2.6 `PSG_CH2`** `#define PSG_CH2 0b01000000`

**20.47.2.7 `PSG_CH3`** `#define PSG_CH3 0b01100000`



**20.47.2.8 PSG\_VOLUME** `#define PSG_VOLUME 0b00010000`

**20.47.2.9 STATF\_INT\_VBL** `#define STATF_INT_VBL 0b10000000`

**20.47.2.10 STATF\_9\_SPR** `#define STATF_9_SPR 0b01000000`

**20.47.2.11 STATF\_SPR\_COLL** `#define STATF_SPR_COLL 0b00100000`

**20.47.2.12 VDP\_REG\_MASK** `#define VDP_REG_MASK 0b10000000`

**20.47.2.13 VDP\_R0** `#define VDP_R0 0b10000000`

**20.47.2.14 R0\_DEFAULT** `#define R0_DEFAULT 0b00000000`

**20.47.2.15 R0\_CB\_OUTPUT** `#define R0_CB_OUTPUT 0b00000000`

**20.47.2.16 R0\_CB\_INPUT** `#define R0_CB_INPUT 0b01000000`

**20.47.2.17 R0\_IE2\_OFF** `#define R0_IE2_OFF 0b00000000`

**20.47.2.18 R0\_IE2** `#define R0_IE2 0b00100000`

**20.47.2.19 R0\_IE1\_OFF** `#define R0_IE1_OFF 0b00000000`

**20.47.2.20 R0\_IE1** `#define R0_IE1 0b00010000`

**20.47.2.21 R0\_SCR\_MODE1** `#define R0_SCR_MODE1 0b00000000`

**20.47.2.22 R0\_SCR\_MODE2** `#define R0_SCR_MODE2 0b00000010`

**20.47.2.23 R0\_SCR\_MODE3** `#define R0_SCR_MODE3 0b00000100`

**20.47.2.24 R0\_ES\_OFF** `#define R0_ES_OFF 0b00000000`

**20.47.2.25 R0\_ES** `#define R0_ES 0b00000001`

**20.47.2.26 VDP\_R1** `#define VDP_R1 0b10000001`

**20.47.2.27 R1\_DEFAULT** `#define R1_DEFAULT 0b10000000`

**20.47.2.28 R1\_DISP\_OFF** `#define R1_DISP_OFF 0b00000000`

**20.47.2.29 R1\_DISP\_ON** `#define R1_DISP_ON 0b01000000`

**20.47.2.30 R1\_IE\_OFF** `#define R1_IE_OFF 0b00000000`

**20.47.2.31 R1\_IE** `#define R1_IE 0b00100000`

**20.47.2.32 R1\_SCR\_MODE1** `#define R1_SCR_MODE1 0b00010000`

**20.47.2.33 R1\_SCR\_MODE2** `#define R1_SCR_MODE2 0b00000000`

**20.47.2.34 R1\_SCR\_MODE3** `#define R1_SCR_MODE3 0b00000000`

**20.47.2.35 R1\_SPR\_8X8** `#define R1_SPR_8X8 0b00000000`

**20.47.2.36 R1\_SPR\_16X16** `#define R1_SPR_16X16 0b00000010`

**20.47.2.37 R1\_SPR\_MAG** `#define R1_SPR_MAG 0b00000001`

**20.47.2.38 R1\_SPR\_MAG\_OFF** `#define R1_SPR_MAG_OFF 0b00000000`

**20.47.2.39 VDP\_R2** `#define VDP_R2 0b10000010`

**20.47.2.40 R2\_MAP\_0x3800** `#define R2_MAP_0x3800 0xFF`

**20.47.2.41 R2\_MAP\_0x3000** `#define R2_MAP_0x3000 0xFD`

**20.47.2.42 R2\_MAP\_0x2800** `#define R2_MAP_0x2800 0xFB`

**20.47.2.43 R2\_MAP\_0x2000** `#define R2_MAP_0x2000 0xF9`

**20.47.2.44 R2\_MAP\_0x1800** `#define R2_MAP_0x1800 0xF7`

**20.47.2.45 R2\_MAP\_0x1000** `#define R2_MAP_0x1000 0xF5`

**20.47.2.46 R2\_MAP\_0x0800** `#define R2_MAP_0x0800 0xF3`

**20.47.2.47 R2\_MAP\_0x0000** `#define R2_MAP_0x0000 0xF1`

**20.47.2.48 VDP\_R3** `#define VDP_R3 0b10000011`

**20.47.2.49 VDP\_R4** `#define VDP_R4 0b10000100`

**20.47.2.50 VDP\_R5** `#define VDP_R5 0b10000101`

**20.47.2.51 R5\_SAT\_0x3F00** `#define R5_SAT_0x3F00 0xFF`

**20.47.2.52 R5\_SAT\_MASK** `#define R5_SAT_MASK 0b10000001`

**20.47.2.53 VDP\_R6** `#define VDP_R6 0b10000110`

**20.47.2.54 R6\_BANK0** `#define R6_BANK0 0xFB`

**20.47.2.55 R6\_DATA\_0x0000** `#define R6_DATA_0x0000 0xFB`

**20.47.2.56 R6\_BANK1** `#define R6_BANK1 0xFF`

**20.47.2.57 R6\_DATA\_0x2000** `#define R6_DATA_0x2000 0xFF`

**20.47.2.58 VDP\_R7** `#define VDP_R7 0b10000111`

**20.47.2.59 VDP\_RBORDER** `#define VDP_RBORDER 0b10000111`

**20.47.2.60 R7\_COLOR\_MASK** `#define R7_COLOR_MASK 0b11110000`

**20.47.2.61 VDP\_R8** `#define VDP_R8 0b10001000`

**20.47.2.62 VDP\_RSCX** `#define VDP_RSCX 0b10001000`

**20.47.2.63 VDP\_R9** `#define VDP_R9 0b10001001`

**20.47.2.64 VDP\_RSCY** `#define VDP_RSCY 0b10001001`

**20.47.2.65 VDP\_R10** `#define VDP_R10 0b10001010`

**20.47.2.66 R10\_INT\_OFF** `#define R10_INT_OFF 0xFF`

**20.47.2.67 R10\_INT EVERY** `#define R10_INT EVERY 0x00`

**20.47.2.68 SYSTEM\_PAL** `#define SYSTEM_PAL 0x00`

**20.47.2.69 SYSTEM\_NTSC** `#define SYSTEM_NTSC 0x01`

**20.47.2.70 VBK\_TILES** `#define VBK_TILES 0`

**20.47.2.71 VBK\_ATTRIBUTES** `#define VBK_ATTRIBUTES 1`

**20.47.2.72 VDP\_SAT\_TERM** `#define VDP_SAT_TERM 0xD0`

**20.47.2.73 DEVICE\_SCREEN\_PX\_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

**20.47.2.74 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 [extern]`

**20.47.3.2 shadow\_VDP\_R1** `UBYTE shadow_VDP_R1 [extern]`

**20.47.3.3 shadow\_VDP\_R2** `UBYTE shadow_VDP_R2 [extern]`

**20.47.3.4 shadow\_VDP\_R3** `UBYTE shadow_VDP_R3 [extern]`

**20.47.3.5 shadow\_VDP\_R4** `UBYTE shadow_VDP_R4` [extern]

**20.47.3.6 shadow\_VDP\_R5** `UBYTE shadow_VDP_R5` [extern]

**20.47.3.7 shadow\_VDP\_R6** `UBYTE shadow_VDP_R6` [extern]

**20.47.3.8 shadow\_VDP\_R7** `UBYTE shadow_VDP_R7` [extern]

**20.47.3.9 shadow\_VDP\_RBORDER** `UBYTE shadow_VDP_RBORDER` [extern]

**20.47.3.10 shadow\_VDP\_R8** `UBYTE shadow_VDP_R8` [extern]

**20.47.3.11 shadow\_VDP\_RSCX** `UBYTE shadow_VDP_RSCX` [extern]

**20.47.3.12 shadow\_VDP\_R9** `UBYTE shadow_VDP_R9` [extern]

**20.47.3.13 shadow\_VDP\_RSCY** `UBYTE shadow_VDP_RSCY` [extern]

**20.47.3.14 shadow\_VDP\_R10** `UBYTE shadow_VDP_R10` [extern]

**20.47.3.15 \_SYSTEM** `const UBYTE _SYSTEM` [extern]

**20.47.3.16 VDP\_ATTR\_SHIFT** `volatile UBYTE VDP_ATTR_SHIFT` [extern]

## 20.48 gbdk-lib/include/nes/hardware.h File Reference

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

### Macros

- `#define __SHADOW_REG` extern volatile `uint8_t`
- `#define __REG(addr)` volatile `__at (addr) uint8_t`
- `#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_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 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 1`
- `#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)`

## Functions

- `__REG (0x2000) PPUCTRL`
- `__REG (0x2001) PPUMASK`
- `__REG (0x2002) PPUSTATUS`
- `__REG (0x2003) OAMADDR`
- `__REG (0x2004) OAMDATA`
- `__REG (0x2005) PPUSCROLL`
- `__REG (0x2006) PPUADDR`
- `__REG (0x2007) PPUDATA`
- `__REG (0x4014) OAMDMA`

## Variables

- `__SHADOW_REG shadow_PPUCTRL`
- `__SHADOW_REG shadow_PPUMASK`
- `__SHADOW_REG bkg_scroll_x`
- `__SHADOW_REG bkg_scroll_y`

### 20.48.1 Detailed Description

Defines that let the NES hardware registers be accessed from C.

### 20.48.2 Macro Definition Documentation

**20.48.2.1 `__SHADOW_REG`** `#define __SHADOW_REG extern volatile uint8_t`

**20.48.2.2 `__REG`** `#define __REG( addr ) volatile __at (addr) uint8_t`

**20.48.2.3 `PPUCTRL_NMI`** `#define PPUCTRL_NMI 0b10000000`

**20.48.2.4 `PPUCTRL_SPR_8X8`** `#define PPUCTRL_SPR_8X8 0b00000000`

**20.48.2.5 PPUCTRL\_SPR\_8X16** `#define PPUCTRL_SPR_8X16 0b00100000`

**20.48.2.6 PPUCTRL\_BG\_CHR** `#define PPUCTRL_BG_CHR 0b00010000`

**20.48.2.7 PPUCTRL\_SPR\_CHR** `#define PPUCTRL_SPR_CHR 0b00001000`

**20.48.2.8 PPUCTRL\_INC32** `#define PPUCTRL_INC32 0b00000100`

**20.48.2.9 PPUMASK\_BLUE** `#define PPUMASK_BLUE 0b10000000`

**20.48.2.10 PPUMASK\_RED** `#define PPUMASK_RED 0b01000000`

**20.48.2.11 PPUMASK\_GREEN** `#define PPUMASK_GREEN 0b00100000`

**20.48.2.12 PPUMASK\_SHOW\_SPR** `#define PPUMASK_SHOW_SPR 0b00010000`

**20.48.2.13 PPUMASK\_SHOW\_BG** `#define PPUMASK_SHOW_BG 0b00001000`

**20.48.2.14 PPUMASK\_SHOW\_SPR\_LC** `#define PPUMASK_SHOW_SPR_LC 0b00000100`

**20.48.2.15 PPUMASK\_SHOW\_BG\_LC** `#define PPUMASK_SHOW_BG_LC 0b00000010`

**20.48.2.16 PPUMASK\_MONOCHROME** `#define PPUMASK_MONOCHROME 0b00000001`

**20.48.2.17 DEVICE\_SCREEN\_X\_OFFSET** `#define DEVICE_SCREEN_X_OFFSET 0`

**20.48.2.18 DEVICE\_SCREEN\_Y\_OFFSET** `#define DEVICE_SCREEN_Y_OFFSET 0`

**20.48.2.19 DEVICE\_SCREEN\_WIDTH** `#define DEVICE_SCREEN_WIDTH 32`

**20.48.2.20 DEVICE\_SCREEN\_HEIGHT** `#define DEVICE_SCREEN_HEIGHT 30`

**20.48.2.21 DEVICE\_SCREEN\_BUFFER\_WIDTH** `#define DEVICE_SCREEN_BUFFER_WIDTH 32`

**20.48.2.22 DEVICE\_SCREEN\_BUFFER\_HEIGHT** `#define DEVICE_SCREEN_BUFFER_HEIGHT 30`

**20.48.2.23** **DEVICE\_SCREEN\_MAP\_ENTRY\_SIZE** `#define DEVICE_SCREEN_MAP_ENTRY_SIZE 1`

**20.48.2.24** **DEVICE\_SPRITE\_PX\_OFFSET\_X** `#define DEVICE_SPRITE_PX_OFFSET_X 0`

**20.48.2.25** **DEVICE\_SPRITE\_PX\_OFFSET\_Y** `#define DEVICE_SPRITE_PX_OFFSET_Y -1`

**20.48.2.26** **DEVICE\_WINDOW\_PX\_OFFSET\_X** `#define DEVICE_WINDOW_PX_OFFSET_X 0`

**20.48.2.27** **DEVICE\_WINDOW\_PX\_OFFSET\_Y** `#define DEVICE_WINDOW_PX_OFFSET_Y 0`

**20.48.2.28** **DEVICE\_SCREEN\_PX\_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

**20.48.2.29** **DEVICE\_SCREEN\_PX\_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

### 20.48.3 Function Documentation

**20.48.3.1** **\_\_REG()** [1/9] `__REG ( 0x2000 )`

**20.48.3.2** **\_\_REG()** [2/9] `__REG ( 0x2001 )`

**20.48.3.3** **\_\_REG()** [3/9] `__REG ( 0x2002 )`

**20.48.3.4** **\_\_REG()** [4/9] `__REG ( 0x2003 )`

**20.48.3.5** **\_\_REG()** [5/9] `__REG ( 0x2004 )`

**20.48.3.6** **\_\_REG()** [6/9] `__REG ( 0x2005 )`

**20.48.3.7** **\_\_REG()** [7/9] `__REG ( 0x2006 )`



**20.48.3.8** `__REG()` [8/9] `__REG (`  
`0x2007 )`

**20.48.3.9** `__REG()` [9/9] `__REG (`  
`0x4014 )`

## 20.48.4 Variable Documentation

**20.48.4.1** `shadow_PPCTRL` `__SHADOW_REG` `shadow_PPCTRL`

**20.48.4.2** `shadow_PPUMASK` `__SHADOW_REG` `shadow_PPUMASK`

**20.48.4.3** `bkg_scroll_x` `__SHADOW_REG` `bkg_scroll_x`

**20.48.4.4** `bkg_scroll_y` `__SHADOW_REG` `bkg_scroll_y`

## 20.49 gbdk-lib/include/sms/hardware.h File Reference

`#include <types.h>`

### Macros

- `#define __BYTES` extern `UBYTE`
- `#define __BYTE_REG` extern volatile `UBYTE`
- `#define GGSTATE_STT` `0b10000000`
- `#define GGSTATE_NJAP` `0b01000000`
- `#define GGSTATE_NNTS` `0b00100000`
- `#define GGEXT_NINIT` `0b10000000`
- `#define SIOCTL_TXFL` `0b00000001`
- `#define SIOCTL_RXRD` `0b00000010`
- `#define SIOCTL_FRER` `0b00000100`
- `#define SIOCTL_INT` `0b00001000`
- `#define SIOCTL_TON` `0b00010000`
- `#define SIOCTL_RON` `0b00100000`
- `#define SIOCTL_BS0` `0b01000000`
- `#define SIOCTL_BS1` `0b10000000`
- `#define SOUNDPAN_TN1R` `0b00000001`
- `#define SOUNDPAN_TN2R` `0b00000010`
- `#define SOUNDPAN_TN3R` `0b00000100`
- `#define SOUNDPAN_NOSR` `0b00001000`
- `#define SOUNDPAN_TN1L` `0b00010000`
- `#define SOUNDPAN_TN2L` `0b00100000`
- `#define SOUNDPAN_TN3L` `0b01000000`
- `#define SOUNDPAN_NOSL` `0b10000000`
- `#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\_TR\_DIR\_IN 0b00000001
- #define JOY\_P1\_TR\_DIR\_OUT 0b00000000
- #define JOY\_P1\_TH\_DIR\_IN 0b00000010
- #define GUN\_P1\_LATCH JOY\_P1\_TH\_DIR\_IN
- #define JOY\_P1\_TH\_DIR\_OUT 0b00000000
- #define JOY\_P2\_TR\_DIR\_IN 0b00000100
- #define JOY\_P2\_TR\_DIR\_OUT 0b00000000
- #define JOY\_P2\_TH\_DIR\_IN 0b00001000
- #define GUN\_P2\_LATCH JOY\_P2\_TH\_DIR\_IN
- #define JOY\_P2\_TH\_DIR\_OUT 0b00000000
- #define JOY\_P1\_TR\_OUT\_HI 0b00010000
- #define JOY\_P1\_TR\_OUT\_LO 0b00000000
- #define JOY\_P1\_TH\_OUT\_HI 0b00100000
- #define JOY\_P1\_TH\_OUT\_LO 0b00000000
- #define JOY\_P2\_TR\_OUT\_HI 0b01000000
- #define JOY\_P2\_TR\_OUT\_LO 0b00000000
- #define JOY\_P2\_TH\_OUT\_HI 0b10000000
- #define JOY\_P2\_TH\_OUT\_LO 0b00000000
- #define JOY\_TH\_HI (JOY\_P1\_TR\_DIR\_IN | JOY\_P1\_TH\_DIR\_OUT | JOY\_P2\_TR\_DIR\_IN | JOY\_P2\_TH\_DIR\_OUT | JOY\_P1\_TR\_OUT\_HI | JOY\_P1\_TH\_OUT\_HI | JOY\_P2\_TR\_OUT\_HI | JOY\_P2\_TH\_OUT\_HI)
- #define JOY\_TH\_LO (JOY\_P1\_TR\_DIR\_IN | JOY\_P1\_TH\_DIR\_OUT | JOY\_P2\_TR\_DIR\_IN | JOY\_P2\_TH\_DIR\_OUT | JOY\_P1\_TR\_OUT\_HI | JOY\_P1\_TH\_OUT\_LO | JOY\_P2\_TR\_OUT\_HI | JOY\_P2\_TH\_OUT\_LO)
- #define PSG\_LATCH 0b10000000
- #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\\_0x1F00](#) 0xBF
- #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\\_MD\\_Z JOY\\_P1\\_UP](#)
- #define [JOY\\_P1\\_DOWN](#) 0b00000010
- #define [JOY\\_P1\\_MD\\_Y JOY\\_P1\\_DOWN](#)
- #define [JOY\\_P1\\_LEFT](#) 0b00000100
- #define [JOY\\_P1\\_MD\\_X JOY\\_P1\\_LEFT](#)
- #define [JOY\\_P1\\_RIGHT](#) 0b00001000
- #define [JOY\\_P1\\_MD\\_MODE JOY\\_P1\\_RIGHT](#)
- #define [JOY\\_P1\\_SW1](#) 0b00010000
- #define [JOY\\_P1\\_TRIGGER JOY\\_P1\\_SW1](#)
- #define [JOY\\_P1\\_MD\\_A JOY\\_P1\\_SW1](#)
- #define [JOY\\_P1\\_SW2](#) 0b00100000
- #define [JOY\\_P1\\_MD\\_START JOY\\_P1\\_SW2](#)
- #define [JOY\\_P2\\_UP](#) 0b01000000
- #define [JOY\\_P2\\_MD\\_Z JOY\\_P2\\_UP](#)
- #define [JOY\\_P2\\_DOWN](#) 0b10000000
- #define [JOY\\_P2\\_MD\\_Y JOY\\_P2\\_DOWN](#)
- #define [JOY\\_P2\\_LEFT](#) 0b00000001

- `#define JOY_P2_MD_X JOY_P2_LEFT`
- `#define JOY_P2_RIGHT 0b00000010`
- `#define JOY_P2_MD_MODE JOY_P2_RIGHT`
- `#define JOY_P2_SW1 0b00000100`
- `#define JOY_P2_TRIGGER JOY_P2_SW1`
- `#define JOY_P2_MD_A JOY_P2_SW1`
- `#define JOY_P2_SW2 0b00001000`
- `#define JOY_P2_MD_START JOY_P2_SW2`
- `#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 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`
- `volatile UBYTE VDP_ATTR_SHIFT`

#### 20.49.1 Detailed Description

Defines that let the SMS/GG hardware registers be accessed from C.

#### 20.49.2 Macro Definition Documentation

**20.49.2.1 `__BYTES`** `#define __BYTES extern UBYTE`

**20.49.2.2 `__BYTE_REG`** `#define __BYTE_REG extern volatile UBYTE`

**20.49.2.3 `GGSTATE_STT`** `#define GGSTATE_STT 0b10000000`

**20.49.2.4 GGSTATE\_NJAP** `#define GGSTATE_NJAP 0b01000000`

**20.49.2.5 GGSTATE\_NNTS** `#define GGSTATE_NNTS 0b00100000`

**20.49.2.6 GGEXT\_NINIT** `#define GGEXT_NINIT 0b10000000`

**20.49.2.7 SIOCTL\_TXFL** `#define SIOCTL_TXFL 0b00000001`

**20.49.2.8 SIOCTL\_RXRD** `#define SIOCTL_RXRD 0b00000010`

**20.49.2.9 SIOCTL\_FRER** `#define SIOCTL_FRER 0b00000100`

**20.49.2.10 SIOCTL\_INT** `#define SIOCTL_INT 0b00001000`

**20.49.2.11 SIOCTL\_TON** `#define SIOCTL_TON 0b00010000`

**20.49.2.12 SIOCTL\_RON** `#define SIOCTL_RON 0b00100000`

**20.49.2.13 SIOCTL\_BS0** `#define SIOCTL_BS0 0b01000000`

**20.49.2.14 SIOCTL\_BS1** `#define SIOCTL_BS1 0b10000000`

**20.49.2.15 SOUNDPAN\_TN1R** `#define SOUNDPAN_TN1R 0b00000001`

**20.49.2.16 SOUNDPAN\_TN2R** `#define SOUNDPAN_TN2R 0b00000010`

**20.49.2.17 SOUNDPAN\_TN3R** `#define SOUNDPAN_TN3R 0b00000100`

**20.49.2.18 SOUNDPAN\_NOSR** `#define SOUNDPAN_NOSR 0b00001000`

**20.49.2.19 SOUNDPAN\_TN1L** `#define SOUNDPAN_TN1L 0b00010000`

**20.49.2.20 SOUNDPAN\_TN2L** `#define SOUNDPAN_TN2L 0b00100000`

**20.49.2.21 SOUNDPAN\_TN3L** `#define SOUNDPAN_TN3L 0b01000000`

**20.49.2.22    SOUNDPAN\_NOSL**    `#define SOUNDPAN_NOSL 0b10000000`

**20.49.2.23    MEMCTL\_JOYON**    `#define MEMCTL_JOYON 0b00000000`

**20.49.2.24    MEMCTL\_JOYOFF**    `#define MEMCTL_JOYOFF 0b00000100`

**20.49.2.25    MEMCTL\_BASEON**    `#define MEMCTL_BASEON 0b00000000`

**20.49.2.26    MEMCTL\_BASEOFF**    `#define MEMCTL_BASEOFF 0b00001000`

**20.49.2.27    MEMCTL\_RAMON**    `#define MEMCTL_RAMON 0b00000000`

**20.49.2.28    MEMCTL\_RAMOFF**    `#define MEMCTL_RAMOFF 0b00010000`

**20.49.2.29    MEMCTL\_CROMON**    `#define MEMCTL_CROMON 0b00000000`

**20.49.2.30    MEMCTL\_CROMOFF**    `#define MEMCTL_CROMOFF 0b00100000`

**20.49.2.31    MEMCTL\_ROMON**    `#define MEMCTL_ROMON 0b00000000`

**20.49.2.32    MEMCTL\_ROMOFF**    `#define MEMCTL_ROMOFF 0b01000000`

**20.49.2.33    MEMCTL\_EXTON**    `#define MEMCTL_EXTON 0b00000000`

**20.49.2.34    MEMCTL\_EXTOFF**    `#define MEMCTL_EXTOFF 0b10000000`

**20.49.2.35    JOY\_P1\_TR\_DIR\_IN**    `#define JOY_P1_TR_DIR_IN 0b00000001`

**20.49.2.36    JOY\_P1\_TR\_DIR\_OUT**    `#define JOY_P1_TR_DIR_OUT 0b00000000`

**20.49.2.37    JOY\_P1\_TH\_DIR\_IN**    `#define JOY_P1_TH_DIR_IN 0b00000010`

**20.49.2.38    GUN\_P1\_LATCH**    `#define GUN_P1_LATCH JOY\_P1\_TH\_DIR\_IN`

**20.49.2.39    JOY\_P1\_TH\_DIR\_OUT**    `#define JOY_P1_TH_DIR_OUT 0b00000000`

**20.49.2.40 JOY\_P2\_TR\_DIR\_IN** `#define JOY_P2_TR_DIR_IN 0b00000100`

**20.49.2.41 JOY\_P2\_TR\_DIR\_OUT** `#define JOY_P2_TR_DIR_OUT 0b00000000`

**20.49.2.42 JOY\_P2\_TH\_DIR\_IN** `#define JOY_P2_TH_DIR_IN 0b00001000`

**20.49.2.43 GUN\_P2\_LATCH** `#define GUN_P2_LATCH JOY_P2_TH_DIR_IN`

**20.49.2.44 JOY\_P2\_TH\_DIR\_OUT** `#define JOY_P2_TH_DIR_OUT 0b00000000`

**20.49.2.45 JOY\_P1\_TR\_OUT\_HI** `#define JOY_P1_TR_OUT_HI 0b00010000`

**20.49.2.46 JOY\_P1\_TR\_OUT\_LO** `#define JOY_P1_TR_OUT_LO 0b00000000`

**20.49.2.47 JOY\_P1\_TH\_OUT\_HI** `#define JOY_P1_TH_OUT_HI 0b00100000`

**20.49.2.48 JOY\_P1\_TH\_OUT\_LO** `#define JOY_P1_TH_OUT_LO 0b00000000`

**20.49.2.49 JOY\_P2\_TR\_OUT\_HI** `#define JOY_P2_TR_OUT_HI 0b01000000`

**20.49.2.50 JOY\_P2\_TR\_OUT\_LO** `#define JOY_P2_TR_OUT_LO 0b00000000`

**20.49.2.51 JOY\_P2\_TH\_OUT\_HI** `#define JOY_P2_TH_OUT_HI 0b10000000`

**20.49.2.52 JOY\_P2\_TH\_OUT\_LO** `#define JOY_P2_TH_OUT_LO 0b00000000`

**20.49.2.53 JOY\_TH\_HI** `#define JOY_TH_HI (JOY_P1_TR_DIR_IN | JOY_P1_TH_DIR_OUT | JOY_P2_TR_DIR_IN | JOY_P2_TH_DIR_OUT | JOY_P1_TR_OUT_HI | JOY_P1_TH_OUT_HI | JOY_P2_TR_OUT_HI | JOY_P2_TH_OUT_HI)`

**20.49.2.54 JOY\_TH\_LO** `#define JOY_TH_LO (JOY_P1_TR_DIR_IN | JOY_P1_TH_DIR_OUT | JOY_P2_TR_DIR_IN | JOY_P2_TH_DIR_OUT | JOY_P1_TR_OUT_HI | JOY_P1_TH_OUT_LO | JOY_P2_TR_OUT_HI | JOY_P2_TH_OUT_LO)`

**20.49.2.55 PSG\_LATCH** `#define PSG_LATCH 0b10000000`

**20.49.2.56 PSG\_CH0** `#define PSG_CH0 0b00000000`

**20.49.2.57 PSG\_CH1** `#define PSG_CH1 0b00100000`

**20.49.2.58 PSG\_CH2** `#define PSG_CH2 0b01000000`

**20.49.2.59 PSG\_CH3** `#define PSG_CH3 0b01100000`

**20.49.2.60 PSG\_VOLUME** `#define PSG_VOLUME 0b00010000`

**20.49.2.61 STATF\_INT\_VBL** `#define STATF_INT_VBL 0b10000000`

**20.49.2.62 STATF\_9\_SPR** `#define STATF_9_SPR 0b01000000`

**20.49.2.63 STATF\_SPR\_COLL** `#define STATF_SPR_COLL 0b00100000`

**20.49.2.64 VDP\_REG\_MASK** `#define VDP_REG_MASK 0b10000000`

**20.49.2.65 VDP\_R0** `#define VDP_R0 0b10000000`

**20.49.2.66 R0\_VSCRL** `#define R0_VSCRL 0b00000000`

**20.49.2.67 R0\_VSCRL\_INH** `#define R0_VSCRL_INH 0b10000000`

**20.49.2.68 R0\_HSCRL** `#define R0_HSCRL 0b00000000`

**20.49.2.69 R0\_HSCRL\_INH** `#define R0_HSCRL_INH 0b01000000`

**20.49.2.70 R0\_NO\_LCB** `#define R0_NO_LCB 0b00000000`

**20.49.2.71 R0\_LCB** `#define R0_LCB 0b00100000`

**20.49.2.72 R0\_IE1\_OFF** `#define R0_IE1_OFF 0b00000000`

**20.49.2.73 R0\_IE1** `#define R0_IE1 0b00010000`

**20.49.2.74 R0\_SS\_OFF** `#define R0_SS_OFF 0b00000000`



**20.49.2.75 R0\_SS** `#define R0_SS 0b00001000`

**20.49.2.76 R0\_DEFAULT** `#define R0_DEFAULT 0b00000110`

**20.49.2.77 R0\_ES\_OFF** `#define R0_ES_OFF 0b00000000`

**20.49.2.78 R0\_ES** `#define R0_ES 0b00000001`

**20.49.2.79 VDP\_R1** `#define VDP_R1 0b10000001`

**20.49.2.80 R1\_DEFAULT** `#define R1_DEFAULT 0b10000000`

**20.49.2.81 R1\_DISP\_OFF** `#define R1_DISP_OFF 0b00000000`

**20.49.2.82 R1\_DISP\_ON** `#define R1_DISP_ON 0b01000000`

**20.49.2.83 R1\_IE\_OFF** `#define R1_IE_OFF 0b00000000`

**20.49.2.84 R1\_IE** `#define R1_IE 0b00100000`

**20.49.2.85 R1\_SPR\_8X8** `#define R1_SPR_8X8 0b00000000`

**20.49.2.86 R1\_SPR\_8X16** `#define R1_SPR_8X16 0b00000010`

**20.49.2.87 VDP\_R2** `#define VDP_R2 0b10000010`

**20.49.2.88 R2\_MAP\_0x3800** `#define R2_MAP_0x3800 0xFF`

**20.49.2.89 R2\_MAP\_0x3000** `#define R2_MAP_0x3000 0xFD`

**20.49.2.90 R2\_MAP\_0x2800** `#define R2_MAP_0x2800 0xFB`

**20.49.2.91 R2\_MAP\_0x2000** `#define R2_MAP_0x2000 0xF9`

**20.49.2.92 R2\_MAP\_0x1800** `#define R2_MAP_0x1800 0xF7`

**20.49.2.93 R2\_MAP\_0x1000** #define R2\_MAP\_0x1000 0xF5

**20.49.2.94 R2\_MAP\_0x0800** #define R2\_MAP\_0x0800 0xF3

**20.49.2.95 R2\_MAP\_0x0000** #define R2\_MAP\_0x0000 0xF1

**20.49.2.96 VDP\_R3** #define VDP\_R3 0b10000011

**20.49.2.97 VDP\_R4** #define VDP\_R4 0b10000100

**20.49.2.98 VDP\_R5** #define VDP\_R5 0b10000101

**20.49.2.99 R5\_SAT\_0x3F00** #define R5\_SAT\_0x3F00 0xFF

**20.49.2.100 R5\_SAT\_0x1F00** #define R5\_SAT\_0x1F00 0xBF

**20.49.2.101 R5\_SAT\_MASK** #define R5\_SAT\_MASK 0b10000001

**20.49.2.102 VDP\_R6** #define VDP\_R6 0b10000110

**20.49.2.103 R6\_BANK0** #define R6\_BANK0 0xFB

**20.49.2.104 R6\_DATA\_0x0000** #define R6\_DATA\_0x0000 0xFB

**20.49.2.105 R6\_BANK1** #define R6\_BANK1 0xFF

**20.49.2.106 R6\_DATA\_0x2000** #define R6\_DATA\_0x2000 0xFF

**20.49.2.107 VDP\_R7** #define VDP\_R7 0b10000111

**20.49.2.108 VDP\_RBORDER** #define VDP\_RBORDER 0b10000111

**20.49.2.109 R7\_COLOR\_MASK** #define R7\_COLOR\_MASK 0b11110000

**20.49.2.110 VDP\_R8** #define VDP\_R8 0b10001000

**20.49.2.111 VDP\_RSCX** `#define VDP_RSCX 0b10001000`

**20.49.2.112 VDP\_R9** `#define VDP_R9 0b10001001`

**20.49.2.113 VDP\_RSCY** `#define VDP_RSCY 0b10001001`

**20.49.2.114 VDP\_R10** `#define VDP_R10 0b10001010`

**20.49.2.115 R10\_INT\_OFF** `#define R10_INT_OFF 0xFF`

**20.49.2.116 R10\_INT EVERY** `#define R10_INT EVERY 0x00`

**20.49.2.117 JOY\_P1\_UP** `#define JOY_P1_UP 0b00000001`

**20.49.2.118 JOY\_P1\_MD\_Z** `#define JOY_P1_MD_Z JOY_P1_UP`

**20.49.2.119 JOY\_P1\_DOWN** `#define JOY_P1_DOWN 0b00000010`

**20.49.2.120 JOY\_P1\_MD\_Y** `#define JOY_P1_MD_Y JOY_P1_DOWN`

**20.49.2.121 JOY\_P1\_LEFT** `#define JOY_P1_LEFT 0b00000100`

**20.49.2.122 JOY\_P1\_MD\_X** `#define JOY_P1_MD_X JOY_P1_LEFT`

**20.49.2.123 JOY\_P1\_RIGHT** `#define JOY_P1_RIGHT 0b00001000`

**20.49.2.124 JOY\_P1\_MD\_MODE** `#define JOY_P1_MD_MODE JOY_P1_RIGHT`

**20.49.2.125 JOY\_P1\_SW1** `#define JOY_P1_SW1 0b00010000`

**20.49.2.126 JOY\_P1\_TRIGGER** `#define JOY_P1_TRIGGER JOY_P1_SW1`

**20.49.2.127 JOY\_P1\_MD\_A** `#define JOY_P1_MD_A JOY_P1_SW1`

**20.49.2.128 JOY\_P1\_SW2** `#define JOY_P1_SW2 0b00100000`

**20.49.2.129 JOY\_P1\_MD\_START** `#define JOY_P1_MD_START JOY_P1_SW2`

**20.49.2.130 JOY\_P2\_UP** `#define JOY_P2_UP 0b01000000`

**20.49.2.131 JOY\_P2\_MD\_Z** `#define JOY_P2_MD_Z JOY_P2_UP`

**20.49.2.132 JOY\_P2\_DOWN** `#define JOY_P2_DOWN 0b10000000`

**20.49.2.133 JOY\_P2\_MD\_Y** `#define JOY_P2_MD_Y JOY_P2_DOWN`

**20.49.2.134 JOY\_P2\_LEFT** `#define JOY_P2_LEFT 0b00000001`

**20.49.2.135 JOY\_P2\_MD\_X** `#define JOY_P2_MD_X JOY_P2_LEFT`

**20.49.2.136 JOY\_P2\_RIGHT** `#define JOY_P2_RIGHT 0b00000010`

**20.49.2.137 JOY\_P2\_MD\_MODE** `#define JOY_P2_MD_MODE JOY_P2_RIGHT`

**20.49.2.138 JOY\_P2\_SW1** `#define JOY_P2_SW1 0b00000100`

**20.49.2.139 JOY\_P2\_TRIGGER** `#define JOY_P2_TRIGGER JOY_P2_SW1`

**20.49.2.140 JOY\_P2\_MD\_A** `#define JOY_P2_MD_A JOY_P2_SW1`

**20.49.2.141 JOY\_P2\_SW2** `#define JOY_P2_SW2 0b00001000`

**20.49.2.142 JOY\_P2\_MD\_START** `#define JOY_P2_MD_START JOY_P2_SW2`

**20.49.2.143 JOY\_RESET** `#define JOY_RESET 0b00010000`

**20.49.2.144 JOY\_P1\_LIGHT** `#define JOY_P1_LIGHT 0b01000000`

**20.49.2.145 JOY\_P2\_LIGHT** `#define JOY_P2_LIGHT 0b10000000`

**20.49.2.146 RAMCTL\_BANK** `#define RAMCTL_BANK 0b00000100`

**20.49.2.147 RAMCTL\_ROM** `#define RAMCTL_ROM 0b00000000`

**20.49.2.148 RAMCTL\_RAM** `#define RAMCTL_RAM 0b00001000`

**20.49.2.149 RAMCTL\_RO** `#define RAMCTL_RO 0b00010000`

**20.49.2.150 RAMCTL\_PROT** `#define RAMCTL_PROT 0b10000000`

**20.49.2.151 VBK\_TILES** `#define VBK_TILES 0`

**20.49.2.152 VBK\_ATTRIBUTES** `#define VBK_ATTRIBUTES 1`

**20.49.2.153 VDP\_SAT\_TERM** `#define VDP_SAT_TERM 0xD0`

**20.49.2.154 DEVICE\_SCREEN\_PX\_WIDTH** `#define DEVICE_SCREEN_PX_WIDTH (DEVICE_SCREEN_WIDTH * 8)`

**20.49.2.155 DEVICE\_SCREEN\_PX\_HEIGHT** `#define DEVICE_SCREEN_PX_HEIGHT (DEVICE_SCREEN_HEIGHT * 8)`

### 20.49.3 Variable Documentation

**20.49.3.1 shadow\_VDP\_R0** `UBYTE shadow_VDP_R0 [extern]`

**20.49.3.2 shadow\_VDP\_R1** `UBYTE shadow_VDP_R1 [extern]`

**20.49.3.3 shadow\_VDP\_R2** `UBYTE shadow_VDP_R2 [extern]`

**20.49.3.4 shadow\_VDP\_R3** `UBYTE shadow_VDP_R3 [extern]`

**20.49.3.5 shadow\_VDP\_R4** `UBYTE shadow_VDP_R4 [extern]`

**20.49.3.6 shadow\_VDP\_R5** `UBYTE shadow_VDP_R5 [extern]`

**20.49.3.7 shadow\_VDP\_R6** `UBYTE shadow_VDP_R6 [extern]`

**20.49.3.8 shadow\_VDP\_R7** `UBYTE shadow_VDP_R7 [extern]`

**20.49.3.9 shadow\_VDP\_RBORDER** `UBYTE shadow_VDP_RBORDER [extern]`

**20.49.3.10 shadow\_VDP\_R8** `UBYTE shadow_VDP_R8 [extern]`

**20.49.3.11 shadow\_VDP\_RSCX** `UBYTE shadow_VDP_RSCX [extern]`

**20.49.3.12 shadow\_VDP\_R9** `UBYTE shadow_VDP_R9 [extern]`

**20.49.3.13 shadow\_VDP\_RSCY** `UBYTE shadow_VDP_RSCY [extern]`

**20.49.3.14 shadow\_VDP\_R10** `UBYTE shadow_VDP_R10 [extern]`

**20.49.3.15 VDP\_ATTR\_SHIFT** `volatile UBYTE VDP_ATTR_SHIFT [extern]`

## 20.50 gbdk-lib/include/gb/hblankcpy.h File Reference

`#include <stdint.h>`

### Functions

- void `hblank_copy_vram` (const `uint8_t` \*sour, `uint8_t` count)
- void `hblank_cpy_vram` (const `uint8_t` \*sour, `uint8_t` count)
- void `hblank_copy` (`uint8_t` \*dest, const `uint8_t` \*sour, `uint16_t` size)

### Variables

- `uint8_t` \* `hblank_copy_destination`

### 20.50.1 Function Documentation

**20.50.1.1 hblank\_copy\_vram()** `void hblank_copy_vram (`  
     const `uint8_t` \* *sour*,  
     `uint8_t` *count* )

HBlank stack copy routine

#### Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>sour</i>  | Source address to copy from      |
| <i>count</i> | Number of 16 byte chunks to copy |

Performs the required STAT\_REG, IE\_REG, IF\_REG manipulation when called and restores STAT\_REG and IE\_REG on exit (unlike `hblank_cpy_vram()`).

Before calling:

- Set the destination using `hblank_copy_destination`
- Interrupts must be disabled

See also

[hblank\\_cpy\\_vram](#), [hblank\\_copy\\_destination](#), [hblank\\_copy](#)

**20.50.1.2 hblank\_cpy\_vram()** `void hblank_cpy_vram (`  
`const uint8\_t * sour,`  
`uint8\_t count )`

HBlank stack copy routine

Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>sour</i>  | Source address to copy from      |
| <i>count</i> | Number of 16 byte chunks to copy |

Unlike [hblank\\_copy\\_vram\(\)](#) does not perform the required STAT\_REG, IE\_REG, IF\_REG manipulation, nor does it restore STAT\_REG and IE\_REG on exit.

Before calling:

- Set the destination using [hblank\\_copy\\_destination](#)
- Interrupts must be properly configured
- Interrupts must be disabled

See also

[hblank\\_copy\\_vram](#), [hblank\\_copy\\_destination](#), [hblank\\_copy](#)

**20.50.1.3 hblank\_copy()** `void hblank_copy (`  
`uint8\_t * dest,`  
`const uint8\_t * sour,`  
`uint16\_t size ) [inline]`

HBlank stack copy routine (must be called with interrupts disabled!)

Parameters

|             |                                                     |
|-------------|-----------------------------------------------------|
| <i>dest</i> | destination pointer                                 |
| <i>sour</i> | source pointer                                      |
| <i>size</i> | number of bytes to copy (rounded to 16-byte chunks) |

Performs a fast vram safe copy of data during HBlank.

## 20.50.2 Variable Documentation

**20.50.2.1 hblank\_copy\_destination** `uint8\_t* hblank_copy_destination [extern]`  
 Destination address for hblank copy routine

## 20.51 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`

### 20.51.1 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.

### 20.51.2 Macro Definition Documentation

#### 20.51.2.1 VECTOR\_STAT `#define VECTOR_STAT 0x48`

Address for the STAT interrupt vector

#### 20.51.2.2 VECTOR\_TIMER `#define VECTOR_TIMER 0x50`

Address for the TIMER interrupt vector

#### 20.51.2.3 VECTOR\_SERIAL `#define VECTOR_SERIAL 0x58`

Address for the SERIAL interrupt vector

#### 20.51.2.4 VECTOR\_JOYPAD `#define VECTOR_JOYPAD 0x60`

Address for the JOYPAD interrupt vector

#### 20.51.2.5 ISR\_VECTOR `#define ISR_VECTOR(`

`ADDR,`

`FUNC ) static const isr_vector_t AT((ADDR)) __ISR_ ## ADDR = {0xc3, (void *)&(FUNC)};`

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: <a href="#">VECTOR_STAT</a> , <a href="#">VECTOR_TIMER</a> , <a href="#">VECTOR_SERIAL</a> , <a href="#">VECTOR_JOYPAD</a> |
| <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](#)

### 20.51.2.6 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: <a href="#">VECTOR_STAT</a> , <a href="#">VECTOR_TIMER</a> , <a href="#">VECTOR_SERIAL</a> , <a href="#">VECTOR_JOYPAD</a> |
| <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](#)

## 20.51.3 Typedef Documentation

**20.51.3.1 `isr_vector_t`** typedef struct [isr\\_vector\\_t](#) [isr\\_vector\\_t](#)

**20.51.3.2 `isr_nested_vector_t`** typedef struct [isr\\_nested\\_vector\\_t](#) [isr\\_nested\\_vector\\_t](#)

## 20.52 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_t` from, `uint8_t` to)
- `uint8_t` `move metasprite_ex` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint8_t` x, `uint8_t` y)
- `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_flipx` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `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_flipy` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `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_flipxy` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `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` `__current_base_prop`
- `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)
hide_sprites_range(hiwater, MAX_HARDWARE_SPRITES);
```

#### 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 (`CGB palette`, `BG/WIN priority`, `Tile VRAM Bank`) will get overwritten.

How to use sprite property flags with metasprites:

- Primary method: Use the `base_prop` parameter for the `move metasprite_*`() functions.

- For more details about the properties on the Game Boy see: <https://gbdev.io/pandocs/OAM.html#byte-3-attributesflags>
- This can be left at zero for defaults
- Various OAMF\_\* flags can be used depending on the platform:
  - \* OAMF\_BANK0, OAMF\_BANK1
  - \* OAMF\_CGB\_PAL0, OAMF\_CGB\_PAL1, OAMF\_CGB\_PAL2, OAMF\_CGB\_PAL3, OAMF\_CGB\_PAL4, OAMF\_CGB\_PAL5, OAMF\_CGB\_PAL6, OAMF\_CGB\_PAL7,
  - \* OAMF\_PAL0, OAMF\_PAL1,
  - \* OAMF\_PALMASK, OAMF\_PRI, OAMF\_XFLIP, OAMF\_YFLIP
- Alternate method: The metasprite structures can have the property flags modified before compilation (such as with `-sp <props>` in the [png2asset](#) tool).

The following functions only support hardware sprite flipping on the Game Boy / Mega Duck and NES. For other consoles which do not have hardware sprite flipping see the cross-platform metasprite example for a workaround (with some performance penalty).

- [move metasprite flipx\(\)](#)
- [move metasprite flipy\(\)](#)
- [move metasprite flipxy\(\)](#)

To test for hardware support see [HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_X](#) and [HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_Y](#). Also see [docs\\_consoles\\_supported\\_list](#) for a brief summary of console capabilities.

### 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_t from,`  
     `uint8_t to )`

Hides all hardware sprites in range from  $\leq X < to$

### Parameters

|             |                                                                  |
|-------------|------------------------------------------------------------------|
| <i>from</i> | start OAM index                                                  |
| <i>to</i>   | finish OAM index (must be $\leq \text{MAX\_HARDWARE\_SPRITES}$ ) |

### See also

[hide\\_sprite](#), [MAX\\_HARDWARE\\_SPRITES](#)

Hides all hardware sprites in range from  $\leq X < to$

### Parameters

|             |                  |
|-------------|------------------|
| <i>from</i> | start OAM index  |
| <i>to</i>   | finish OAM index |

**20.52.7.2 [move metasprite\\_ex\(\)](#)** `uint8_t move metasprite_ex (`  
     `const metasprite_t * metasprite,`  
     `uint8_t base_tile,`  
     `uint8_t base_prop,`  
     `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_prop</i>   | Base sprite property flags (can be used to set palette, etc)          |
| <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

**20.52.7.3 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]`

Obsolete. This function has been replaced by [move\\_metasprite\\_ex\(\)](#)

**20.52.7.4 move\_metasprite\_flipx()** `uint8_t move_metasprite_flipx (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint8_t x,`  
`uint8_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by X (horizontally)**

## 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_prop</i>   | Base sprite property flags (can be used to set palette, etc)          |
| <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 by X (horizontally).

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Note: Overwrites OAM sprite properties (such as CGB palette), see [Metaspprites 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\(\)](#)

**20.52.7.5 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]`

Obsolete. This function has been replaced by [move\\_metasprite\\_flipx\(\)](#)

**20.52.7.6 move metasprite flipy()** `uint8_t move metasprite flipy (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint8_t x,`  
`uint8_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by Y (vertically)**

#### 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_prop</i>   | Base sprite property flags (can be used to set palette, etc)          |
| <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 by Y (vertically).

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\(\)](#)

**20.52.7.7 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]`

Obsolete. This function has been replaced by [move metasprite flipy\(\)](#)

**20.52.7.8 move metasprite flipxy()** `uint8_t move metasprite flipxy (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint8_t x,`  
`uint8_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by X and Y (horizontally and vertically)**

#### 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_prop</i>   | Base sprite property flags (can be used to set palette, etc)     |
| <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 by X and Y (horizontally and vertically).  
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\(\)](#)

**20.52.7.9 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]`

Obsolete. This function has been replaced by [move\\_metasprite\\_flipxy\(\)](#)

**20.52.7.10 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 [extern]`

**20.52.8.2 \_\_current\_base\_tile** `uint8_t __current_base_tile [extern]`

**20.52.8.3** `__current_base_prop` `uint8_t __current_base_prop` [extern]

**20.52.8.4** `__render_shadow_OAM` `uint8_t __render_shadow_OAM` [extern]

## 20.53 gbdk-lib/include/gbdk/metasprites.h File Reference

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

## 20.54 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_t` from, `uint8_t` to) `Z88DK_CALLEE PRESERVES_REGS(iyh`
- `uint8_t` `move_metasprite_ex` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint8_t` x, `uint8_t` y)
- `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`

#### 20.54.1 Macro Definition Documentation

**20.54.1.1** `metasprite_end` `#define metasprite_end -128`



**20.54.1.2 METASPR\_ITEM** `#define METASPR_ITEM(`

```

    dy,
    dx,
    dt,
    a ) { (dy), (dx), (dt), (a) }
```

**20.54.1.3 METASPR\_TERM** `#define METASPR_TERM {metasprite_end}`**20.54.2 Typedef Documentation****20.54.2.1 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>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.

**20.54.3 Function Documentation****20.54.3.1 hide\_sprites\_range()** `void hide_sprites_range (`

```

    uint8_t from,
    uint8_t to )
```

Hides all hardware sprites in range from  $\leq X < to$

## Parameters

|             |                  |
|-------------|------------------|
| <i>from</i> | start OAM index  |
| <i>to</i>   | finish OAM index |

**20.54.3.2 move\_metasprite\_ex()** `uint8_t move_metasprite_ex (`

```

    const metasprite_t * metasprite,
    uint8_t base_tile,
    uint8_t base_prop,
    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) |
|-------------------|-----------------------------------------------------------------------|

## Parameters

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| <i>base_tile</i>   | Number of the first tile where the metasprite's tiles start      |
| <i>base_prop</i>   | Base sprite property flags (unused on this platform)             |
| <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

**20.54.3.3 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]`

Obsolete

**20.54.3.4 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;`

**20.54.4 Variable Documentation**

**20.54.4.1 \_\_current\_metasprite** `const void* __current_metasprite [extern]`

**20.54.4.2 \_\_current\_base\_tile** `uint8_t __current_base_tile [extern]`

**20.54.4.3 \_\_render\_shadow\_OAM** `uint8_t __render_shadow_OAM [extern]`

**20.54.4.4** `iy1` `uint8_t` `iy1`**Initial value:**

```
{
    __asm__("ei")
}
```

**20.55 gbdk-lib/include/nes/metasprites.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_t` from, `uint8_t` to) `OLDCALL`
- `uint8_t` `move_metasprite_ex` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_prop`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_flipx` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_prop`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_vflip` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_flipy` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_prop`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_hflip` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_flipxy` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_prop`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_t` `y`)
- `uint8_t` `move_metasprite_hvflip` (const `metasprite_t` \*`metasprite`, `uint8_t` `base_tile`, `uint8_t` `base_sprite`, `int16_t` `x`, `int16_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` `__current_base_prop`
- `uint8_t` `__render_shadow_OAM`

**20.55.1 Detailed Description****20.55.2 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.

See the main [metasprite docs](#) under the game Boy platform for additional details.

### 20.55.3 Macro Definition Documentation

**20.55.3.1 metasprite\_end** `#define metasprite_end -128`

**20.55.3.2 METASPR\_ITEM** `#define METASPR_ITEM(  
     dy,  
     dx,  
     dt,  
     a ) { (dy), (dx), (dt), (a) }`

**20.55.3.3 METASPR\_TERM** `#define METASPR_TERM {metasprite_end}`

### 20.55.4 Typedef Documentation

**20.55.4.1 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.

### 20.55.5 Function Documentation

**20.55.5.1 hide\_sprites\_range()** `void hide_sprites_range (  
     uint8_t from,  
     uint8_t to )`

Hides all hardware sprites in range from  $\leq X < to$

#### Parameters

|             |                  |
|-------------|------------------|
| <i>from</i> | start OAM index  |
| <i>to</i>   | finish OAM index |

**20.55.5.2 move\_metasprite\_ex()** `uint8_t move_metasprite_ex (  
     const metasprite_t * metasprite,  
     uint8_t base_tile,  
     uint8_t base_prop,  
     uint8_t base_sprite,`

```

    int16_t x,
    int16_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_prop</i>   | Base sprite property flags                                            |
| <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 [Metaspprites and sprite properties](#).

#### Returns

Number of hardware sprites used to draw this metasprite

**20.55.5.3 move\_metasprite()** `uint8_t move_metasprite (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Obsolete. Replaced by [move\\_metasprite\\_ex\(\)](#)

**20.55.5.4 move\_metasprite\_flipx()** `uint8_t move_metasprite_flipx (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Moves metasprite to the absolute position **x** and **y**, **flipped by X (horizontally)**

#### 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_prop</i>   | Base sprite property flags                                            |
| <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 by X (horizontally).

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\(\)](#)

**20.55.5.5 move metasprite\_vflip()** `uint8_t move metasprite_vflip (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Obsolete. Replaced by [move metasprite\\_flipx\(\)](#)

**20.55.5.6 move metasprite\_flipy()** `uint8_t move metasprite_flipy (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by Y (vertically)**

#### 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_prop</i>   | Base sprite property flags                                            |
| <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 by Y (vertically).

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\(\)](#)

**20.55.5.7 move\_metasprite\_hflip()** `uint8_t move_metasprite_hflip (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Obsolete. Replaced by [move\\_metasprite\\_flipy\(\)](#)

**20.55.5.8 move\_metasprite\_flipxy()** `uint8_t move_metasprite_flipxy (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by X and Y (horizontally and vertically)**

#### 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_prop</i>   | Base sprite property flags                                            |
| <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 by X and Y (horizontally and vertically).

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\(\)](#)

**20.55.5.9 move\_metasprite\_hvflip()** `uint8_t move_metasprite_hvflip (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_sprite,`  
`int16_t x,`  
`int16_t y ) [inline]`

Obsolete. Replaced by [move\\_metasprite\\_flipy\(\)](#)

**20.55.5.10 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;`

## 20.55.6 Variable Documentation

**20.55.6.1** `__current_metasprite` `const void* __current_metasprite` [extern]

**20.55.6.2** `__current_base_tile` `uint8_t __current_base_tile` [extern]

**20.55.6.3** `__current_base_prop` `uint8_t __current_base_prop` [extern]

**20.55.6.4** `__render_shadow_OAM` `uint8_t __render_shadow_OAM` [extern]

## 20.56 gbdk-lib/include/sms/metasprires.h File Reference

```
#include <sms/sms.h>
#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_t` from, `uint8_t` to) `PRESERVES_REGS(iyh`
- `uint8_t` `move_metasprite_ex` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint16_t` x, `uint16_t` y)
- `uint8_t` `move_metasprite` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_sprite, `uint16_t` x, `uint16_t` y)
- `uint8_t` `move_metasprite_flipx` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint16_t` x, `uint16_t` y)
- `uint8_t` `move_metasprite_flipy` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint16_t` x, `uint16_t` y)



- `uint8_t move_metasprite_flipxy` (const `metasprite_t` \*metasprite, `uint8_t` base\_tile, `uint8_t` base\_prop, `uint8_t` base\_sprite, `uint16_t` x, `uint16_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 void `iyf`

### 20.56.1 Detailed Description

#### 20.56.2 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.

See the main [metasprite docs](#) under the game Boy platform for additional details.

#### 20.56.3 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.

See the main [metasprite docs](#) under the game Boy platform for additional details.

### 20.56.4 Macro Definition Documentation

**20.56.4.1 metasprite\_end** `#define metasprite_end -128`

**20.56.4.2 METASPR\_ITEM** `#define METASPR_ITEM(`  
`dy,`  
`dx,`  
`dt,`  
`a ) { (dy), (dx), (dt) }`

**20.56.4.3 METASPR\_TERM** `#define METASPR_TERM {metasprite_end}`

### 20.56.5 Typedef Documentation

**20.56.5.1 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             |

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.

## 20.56.6 Function Documentation

**20.56.6.1 hide\_sprites\_range()** void hide\_sprites\_range (   
     uint8\_t from,   
     uint8\_t to )

Hides all hardware sprites in range from  $\leq X < to$

### Parameters

|             |                  |
|-------------|------------------|
| <i>from</i> | start OAM index  |
| <i>to</i>   | finish OAM index |

**20.56.6.2 move\_metasprite\_ex()** uint8\_t move\_metasprite\_ex (   
     const metasprite\_t \* metasprite,   
     uint8\_t base\_tile,   
     uint8\_t base\_prop,   
     uint8\_t base\_sprite,   
     uint16\_t x,   
     uint16\_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_prop</i>   | Base sprite property flags (unused on this platform)                  |
| <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

**20.56.6.3 move\_metasprite()** uint8\_t move\_metasprite (   
     const metasprite\_t \* metasprite,   
     uint8\_t base\_tile,   
     uint8\_t base\_sprite,   
     uint16\_t x,   
     uint16\_t y ) [inline]

Obsolete. This function has been replaced by [move\\_metasprite\\_ex\(\)](#)

**20.56.6.4 move\_metaspprite\_flipx()** `uint8_t move_metaspprite_flipx (`  
`const metaspprite_t * metaspprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint16_t x,`  
`uint16_t y ) [inline]`

Moves metaspprite to the absolute position x and y, **flipped by X (horizontally)**

#### Parameters

|                    |                                                                        |
|--------------------|------------------------------------------------------------------------|
| <i>metaspprite</i> | Pointer to the first struct of the metaspprite (for the desired frame) |
| <i>base_tile</i>   | Number of the first tile where the metaspprite's tiles start           |
| <i>base_prop</i>   | Base sprite property flags (unused on this platform)                   |
| <i>base_sprite</i> | Number of the first hardware sprite to be used by the metaspprite      |
| <i>x</i>           | Absolute x coordinate of the sprite                                    |
| <i>y</i>           | Absolute y coordinate of the sprite                                    |

Same as [move\\_metaspprite\(\)](#), but with the metaspprite flipped by X (horizontally).

Sets:

- `__current_metaspprite = metaspprite;`
- `__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 metaspprite

#### See also

[move\\_metaspprite\(\)](#)

**20.56.6.5 move\_metaspprite\_flipy()** `uint8_t move_metaspprite_flipy (`  
`const metaspprite_t * metaspprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint16_t x,`  
`uint16_t y ) [inline]`

Moves metaspprite to the absolute position x and y, **flipped by Y (vertically)**

#### Parameters

|                    |                                                                        |
|--------------------|------------------------------------------------------------------------|
| <i>metaspprite</i> | Pointer to the first struct of the metaspprite (for the desired frame) |
| <i>base_tile</i>   | Number of the first tile where the metaspprite's tiles start           |
| <i>base_prop</i>   | Base sprite property flags (unused on this platform)                   |
| <i>base_sprite</i> | Number of the first hardware sprite to be used by the metaspprite      |
| <i>x</i>           | Absolute x coordinate of the sprite                                    |
| <i>y</i>           | Absolute y coordinate of the sprite                                    |

Same as [move\\_metaspprite\(\)](#), but with the metaspprite flipped by Y (vertically).

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\(\)](#)

**20.56.6.6 move\_metasprite\_flipxy()** `uint8_t move_metasprite_flipxy (`  
`const metasprite_t * metasprite,`  
`uint8_t base_tile,`  
`uint8_t base_prop,`  
`uint8_t base_sprite,`  
`uint16_t x,`  
`uint16_t y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped by X and Y (horizontally and vertically)**

#### 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_prop</i>   | Base sprite property flags (unused on this platform)                  |
| <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 by X and Y (horizontally and vertically).  
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\(\)](#)

**20.56.6.7 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;`

### 20.56.7 Variable Documentation

**20.56.7.1** `__current metasprite` `const void* __current metasprite` [extern]

**20.56.7.2** `__current_base_tile` `uint8_t __current_base_tile` [extern]

**20.56.7.3** `__render_shadow_OAM` `uint8_t __render_shadow_OAM` [extern]

**20.56.7.4** `iyi` `void iyi`

## 20.57 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` (void) `OLDCALL PRESERVES_REGS(b`
- `void sgb_transfer` (uint8\_t \*packet) `OLDCALL PRESERVES_REGS(b`

## Variables

- [uint8\\_t c](#)

### 20.57.1 Detailed Description

Super Gameboy definitions.

See the example SGB project for additional details.

### 20.57.2 Macro Definition Documentation

#### 20.57.2.1 SGB\_PAL\_01 `#define SGB_PAL_01 0x00U`

SGB Command: Set SGB Palettes 0 & 1

#### 20.57.2.2 SGB\_PAL\_23 `#define SGB_PAL_23 0x01U`

SGB Command: Set SGB Palettes 2 & 3

#### 20.57.2.3 SGB\_PAL\_03 `#define SGB_PAL_03 0x02U`

SGB Command: Set SGB Palettes 0 & 3

#### 20.57.2.4 SGB\_PAL\_12 `#define SGB_PAL_12 0x03U`

SGB Command: Set SGB Palettes 1 & 2

#### 20.57.2.5 SGB\_ATTR\_BLK `#define SGB_ATTR_BLK 0x04U`

SGB Command: Set color attributes for rectangular regions

#### 20.57.2.6 SGB\_ATTR\_LIN `#define SGB_ATTR_LIN 0x05U`

SGB Command: Set color attributes for horizontal or vertical character lines

#### 20.57.2.7 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

#### 20.57.2.8 SGB\_ATTR\_CHR `#define SGB_ATTR_CHR 0x07U`

SGB Command: Set color attributes for separate charactersSet SGB Palette 0,1 Data

#### 20.57.2.9 SGB\_SOUND `#define SGB_SOUND 0x08U`

SGB Command: Start and stop a internal sound effect, and sounds using internal tone data

#### 20.57.2.10 SGB\_SOU\_TRN `#define SGB_SOU_TRN 0x09U`

SGB Command: Transfer sound code or data to the SNES APU RAM

#### 20.57.2.11 SGB\_PAL\_SET `#define SGB_PAL_SET 0x0AU`

SGB Command: Apply (previously transferred) SGB system color palettes to actual SNES palettes

#### 20.57.2.12 SGB\_PAL\_TRN `#define SGB_PAL_TRN 0x0BU`

SGB Command: Transfer palette data into SGB system color palettes

#### 20.57.2.13 SGB\_ATTRC\_EN `#define SGB_ATTRC_EN 0x0CU`

SGB Command: Enable/disable Attraction mode. It is enabled by default

#### 20.57.2.14 SGB\_TEST\_EN `#define SGB_TEST_EN 0x0DU`

SGB Command: Enable/disable test mode for "SGB-CPU variable clock speed function"

**20.57.2.15 SGB\_ICON\_EN** `#define SGB_ICON_EN 0x0EU`

SGB Command: Enable/disable ICON functionality

**20.57.2.16 SGB\_DATA\_SND** `#define SGB_DATA_SND 0x0FU`

SGB Command: Write one or more bytes into SNES Work RAM

**20.57.2.17 SGB\_DATA\_TRN** `#define SGB_DATA_TRN 0x10U`

SGB Command: Transfer code or data into SNES RAM

**20.57.2.18 SGB\_MLT\_REQ** `#define SGB_MLT_REQ 0x11U`

SGB Command: Request multiplayer mode (input from more than one joypad)

**20.57.2.19 SGB\_JUMP** `#define SGB_JUMP 0x12U`

SGB Command: Set the SNES program counter and NMI (vblank interrupt) handler to specific addresses

**20.57.2.20 SGB\_CHR\_TRN** `#define SGB_CHR_TRN 0x13U`

SGB Command: Transfer tile data (characters) to SNES Tile memory

**20.57.2.21 SGB\_PCT\_TRN** `#define SGB_PCT_TRN 0x14U`

SGB Command: Transfer tile map and palette data to SNES BG Map memory

**20.57.2.22 SGB\_ATTR\_TRN** `#define SGB_ATTR_TRN 0x15U`

SGB Command: Transfer data to (color) Attribute Files (ATFs) in SNES RAM

**20.57.2.23 SGB\_ATTR\_SET** `#define SGB_ATTR_SET 0x16U`

SGB Command: Transfer attributes from (color) Attribute Files (ATF) to the Game Boy window

**20.57.2.24 SGB\_MASK\_EN** `#define SGB_MASK_EN 0x17U`

SGB Command: Modify Game Boy window mask settings

**20.57.2.25 SGB\_OBJ\_TRN** `#define SGB_OBJ_TRN 0x18U`

SGB Command: Transfer OBJ attributes to SNES OAM memory

## 20.57.3 Function Documentation

**20.57.3.1 sgb\_check()** `uint8_t sgb_check (`  
`void )`

Returns a non-zero value if running on a Super GameBoy

Since [sgb\\_check\(\)](#) uses [sgb\\_transfer\(\)](#), the same delay at startup requirement applies to ensure correct operation on PAL SNES. See [sgb\\_transfer\(\)](#) for details.

**20.57.3.2 sgb\_transfer()** `void sgb_transfer (`  
`uint8_t * packet )`

Transfer a SGB packet

### Parameters

|               |                                         |
|---------------|-----------------------------------------|
| <i>packet</i> | 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\(\)](#)

## 20.57.4 Variable Documentation

### 20.57.4.1 `c` `void c`

## 20.58 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](#) (void) [OLDCALL](#)
- uint8\_t [posy](#) (void) [OLDCALL](#)
- void [setchar](#) (char c) [OLDCALL](#)
- void [cls](#) (void)

### 20.58.1 Detailed Description

Console functions that work like Turbo C's.  
The font is 8x8, making the screen 20x18 characters.

### 20.58.2 Function Documentation

#### 20.58.2.1 `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\(\)](#)

#### 20.58.2.2 `posx()` `uint8_t posx (` `void )`

Returns the current X position of the cursor.

See also

[gotoxy\(\)](#)



**20.58.2.3** `posy()` `uint8_t posy (`  
`void )`

Returns the current Y position of the cursor.

See also

[gotoxy\(\)](#)

**20.58.2.4** `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\(\)](#)

**20.58.2.5** `cls()` `void cls (`  
`void )`

Clears the screen

## 20.59 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 (void)`
- `uint32_t to_far_ptr (void *ofs, uint16_t seg)`

### Variables

- volatile `FAR_PTR __call_banked_ptr`
- volatile `void * __call_banked_addr`
- volatile `uint8_t __call_banked_bank`

### 20.59.1 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)

### 20.59.2 Macro Definition Documentation

**20.59.2.1 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`)

**20.59.2.2 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 <code>FAR_PTR</code> ) |
|------------|--------------------------------------------|

#### Returns

Segment (Bank) of the far pointer (type `uint16_t`)

**20.59.2.3 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 <code>FAR_PTR</code> ) |
|------------|--------------------------------------------|

#### Returns

Offset (address) of the far pointer (type `void *`)

**20.59.2.4 FAR\_FUNC** `#define FAR_FUNC(  
 ptr,`

```
typ ) ((typ) ((union __far_ptr *) &ptr) -> segfn.fn))
```

```
20.59.2.5 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 <a href="#">FAR_PTR</a> ) |
| <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)

20.59.3 Typedef Documentation

```
20.59.3.1 FAR_PTR typedef uint32_t FAR_PTR  
Type for storing a FAR_PTR
```

20.59.4 Function Documentation

```
20.59.4.1 __call_banked() void __call_banked (  
    void )
```

```
20.59.4.2 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](#))

## 20.59.5 Variable Documentation

**20.59.5.1** `__call_banked_ptr` volatile [FAR\\_PTR](#) `__call_banked_ptr` [extern]

**20.59.5.2** `__call_banked_addr` volatile void\* `__call_banked_addr` [extern]

**20.59.5.3** `__call_banked_bank` volatile [uint8\\_t](#) `__call_banked_bank` [extern]

## 20.60 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](#) (void)
- [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](#) []

### 20.60.1 Detailed Description

Multiple font support for the GameBoy Michael Hope, 1999 [michaelh@earthling.net](mailto:michaelh@earthling.net)

## 20.60.2 Macro Definition Documentation

### 20.60.2.1 FONT\_256ENCODING

Various flags in the font header.

### 20.60.2.2 FONT\_128ENCODING

### 20.60.2.3 FONT NOENCODING

#### 20.60.2.4 FONT COMPRESSED `#define FONT COMPRESSED 4`

### 20.60.3 Typedef Documentation

### 20.60.3.1 font\_t

font\_t is a handle to a font loaded by `font_load()`. It can be used with `font_set()`

### 20.60.3.2 mfont\_handle typedef struct sfont\_handle mfont\_handle

## Internal representation of a font. What a font really is

### 20.60.3.3 pmfont handle

#### 20.60.4 Function Documentation

```
20.60.4.1 font_init() void font_init (
                        void )
```

Initializes the font system. Should be called before other font functions.

#### 20.60.4.2 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 gbk font) |
|-------------|------------------------------------------------|

## Returns

Handle to the loaded font, which can be used with `font.set()`

### See also

font init(), font set(), List of gbdk fonts

### 20.60.4.3 font\_set()

```
font_t font_set (
    font_t font_handle )
```

Set the current font.

#### Parameters

|                          |                                                          |
|--------------------------|----------------------------------------------------------|
| <code>font_handle</code> | handle of a font returned by <a href="#">font_load()</a> |
|--------------------------|----------------------------------------------------------|

#### Returns

The previously used font handle.

#### See also

[font\\_init\(\)](#), [font\\_load\(\)](#)

**20.60.4.4 font\_color()** `void font_color (`  
    `uint8_t forecolor,`  
    `uint8_t backcolor )`

Set the current **foreground** colour (for pixels), **background** colour

## 20.61 gbdk-lib/include/gbdk/gbdk-lib.h File Reference

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

### 20.61.1 Detailed Description

Settings for the greater library system.

## 20.62 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)`

### 20.62.1 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.

### 20.62.2 Macro Definition Documentation

**20.62.2.1 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

|                |                                       |
|----------------|---------------------------------------|
| <i>VARNAME</i> | Name of the variable used with INCBIN |
|----------------|---------------------------------------|

An entry is created for the variable and it's size variable.

[INCBIN\(\)](#), [INCBIN\\_SIZE\(\)](#)

**20.62.2.2 INCBIN\_SIZE** `#define INCBIN_SIZE(  
VARNAME ) ( (uint16_t) & __size_ ## VARNAME )`

Obtains the **size in bytes** 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\(\)](#)

**20.62.2.3 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\(\)](#)

**20.62.2.4 INCBIN** `#define INCBIN(  
VARNAME,  
FILEPATH )`

## Value:

```
void __func_ ## VARNAME(void) __banked __naked { \
__asm \
_ ## VARNAME:: \
1$: \
.incbn FILEPATH \
2$: \
__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.

The [INCBIN\(\)](#) macro will declare the [BANK\(\)](#) and [INCBIN\\_SIZE\(\)](#) helper symbols. Then if [INCBIN\\_EXTERN\(\)](#) is used in the header then those helper macros can be used in the application code.

- [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 accessible there.

## 20.63 gbdk-lib/include/gbdk/platform.h File Reference

```
#include <gb/gb.h>
#include <gb/cgb.h>
#include <gb/sgb.h>
```

## 20.64 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)
- [uint8\\_t rle\\_decompress](#) (void \*dest, [uint8\\_t](#) len)

#### 20.64.1 Detailed Description

Decompressor for RLE encoded data

Decompresses data which has been compressed with [gbcompress](#) using the `--alg=rle` argument.

#### 20.64.2 Macro Definition Documentation

##### 20.64.2.1 RLE\_STOP `#define RLE_STOP 0`

#### 20.64.3 Function Documentation

##### 20.64.3.1 [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](#)

##### 20.64.3.2 [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](#)

## 20.65 gbdk-lib/include/gbdk/version.h File Reference

## Macros

- `#define __GBDK_VERSION 430`

### 20.65.1 Macro Definition Documentation

**20.65.1.1** `__GBDK_VERSION` `#define __GBDK_VERSION 430`

## 20.66 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`

### 20.66.1 Macro Definition Documentation

**20.66.1.1** `CHAR_BIT` `#define CHAR_BIT 8 /* bits in a char */`

**20.66.1.2 SCHAR\_MAX** `#define SCHAR_MAX 127`

**20.66.1.3 SCHAR\_MIN** `#define SCHAR_MIN -128`

**20.66.1.4 UCHAR\_MAX** `#define UCHAR_MAX 0xff`

**20.66.1.5 CHAR\_MAX** `#define CHAR_MAX SCHAR\_MAX`

**20.66.1.6 CHAR\_MIN** `#define CHAR_MIN SCHAR\_MIN`

**20.66.1.7 INT\_MIN** `#define INT_MIN (-32767 - 1)`

**20.66.1.8 INT\_MAX** `#define INT_MAX 32767`

**20.66.1.9 SHRT\_MAX** `#define SHRT_MAX INT\_MAX`

**20.66.1.10 SHRT\_MIN** `#define SHRT_MIN INT\_MIN`

**20.66.1.11 UINT\_MAX** `#define UINT_MAX 0xffff`

**20.66.1.12 UINT\_MIN** `#define UINT_MIN 0`

**20.66.1.13 USHRT\_MAX** `#define USHRT_MAX UINT\_MAX`

**20.66.1.14 USHRT\_MIN** `#define USHRT_MIN UINT\_MIN`

**20.66.1.15 LONG\_MIN** `#define LONG_MIN (-2147483647L-1)`

**20.66.1.16 LONG\_MAX** `#define LONG_MAX 2147483647L`

**20.66.1.17 ULONG\_MAX** `#define ULONG_MAX 0xffffffff`

**20.66.1.18 ULONG\_MIN** `#define ULONG_MIN 0`

## 20.67 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 [SYSTEM\\_60HZ](#) 0x00
- #define [SYSTEM\\_50HZ](#) 0x01
- #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\\_BANK](#) 0x01U
- #define [S\\_FLIPX](#) 0x02U
- #define [S\\_FLIPY](#) 0x04U
- #define [S\\_PALETTE](#) 0x08U
- #define [S\\_PRIORITY](#) 0x10U
- #define [S\\_PAL\(n\)](#) (((n) & 0x01U) << 3)
- #define [\\_\\_WRITE\\_VDP\\_REG\\_UNSAFE](#)(REG, v) shadow\_##REG=(v),VDP\_CMD=(shadow\_##REG),VDP\_CMD←\_CMD=REG
- #define [\\_\\_WRITE\\_VDP\\_REG](#)(REG, v) shadow\_##REG=(v);\_\_asm\_\_("di");VDP\_CMD=(shadow\_←\_##REG);VDP\_CMD=REG;\_\_asm\_\_("ei")
- #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 SET_BORDER_COLOR(C) __WRITE_VDP_REG(VDP_R7, ((C) | 0xf0u))`
- `#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 DIV_REG get_r_reg()`
- `#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_CONTROL&(~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 HARDWARE_SPRITE_CAN_FLIP_X 0`
- `#define HARDWARE_SPRITE_CAN_FLIP_Y 0`
- `#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 (void) OLDCALL`
- `uint8_t get_system (void)`
- `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)
- void [move\\_bkg](#) (uint8\_t x, uint8\_t y)
- void [scroll\\_bkg](#) (uint8\_t x, uint8\_t y)
- void [vsync](#) (void) PRESERVES\_REGS(b)
- void [wait\\_vbl\\_done](#) (void) PRESERVES\_REGS(b)
- void [display\\_off](#) (void)
- void [refresh\\_OAM](#) (void)
- [uint8\\_t get\\_r\\_reg](#) (void) PRESERVES\_REGS(b)
- void [SWITCH\\_ROM](#) (uint8\_t bank) Z88DK\_FASTCALL PRESERVES\_REGS(b)
- void [delay](#) (uint16\_t d) Z88DK\_FASTCALL
- [uint8\\_t joypad](#) (void) OLDDCALL PRESERVES\_REGS(b)
- [uint8\\_t waitpad](#) (uint8\_t mask) Z88DK\_FASTCALL PRESERVES\_REGS(b)
- void [waitpadup](#) (void) 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 [enable\\_interrupts](#) (void) PRESERVES\_REGS(a)
- void [disable\\_interrupts](#) (void) PRESERVES\_REGS(a)
- void [set\\_default\\_palette](#) (void)
- void [cpu\\_fast](#) (void)
- 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, const [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

- const uint8\_t `_SYSTEM`
- void `c`
- void `d`
- void `e`
- void `iyh`
- void `iyi`
- void `h`
- void `l`
- volatile uint16\_t `sys_time`
- volatile uint8\_t `current_bank`
- void `b`
- 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`

### 20.67.1 Detailed Description

MSX specific functions.

### 20.67.2 Macro Definition Documentation

#### 20.67.2.1 MSX `#define MSX`

#### 20.67.2.2 SYSTEM\_60HZ `#define SYSTEM_60HZ 0x00`

#### 20.67.2.3 SYSTEM\_50HZ `#define SYSTEM_50HZ 0x01`

**20.67.2.4 VBK\_REG** `#define VBK_REG VDP_ATTR_SHIFT`

**20.67.2.5 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](#)

**20.67.2.6 J\_DOWN** `#define J_DOWN 0b01000000`

**20.67.2.7 J\_LEFT** `#define J_LEFT 0b00010000`

**20.67.2.8 J\_RIGHT** `#define J_RIGHT 0b10000000`

**20.67.2.9 J\_A** `#define J_A 0b00000001`

**20.67.2.10 J\_B** `#define J_B 0b00000100`

**20.67.2.11 J\_SELECT** `#define J_SELECT 0b00001000`

**20.67.2.12 J\_START** `#define J_START 0b00000010`

**20.67.2.13 M\_TEXT\_OUT** `#define M_TEXT_OUT 0x02U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

**20.67.2.14 M\_TEXT\_INOUT** `#define M_TEXT_INOUT 0x03U`

**20.67.2.15 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\(\)](#)

**20.67.2.16 M\_NO\_INTERP** `#define M_NO_INTERP 0x08U`  
Set this to disable interpretation

See also

[mode\(\)](#)

**20.67.2.17 S\_BANK** `#define S_BANK 0x01U`  
The ninth bit of the tile id

**20.67.2.18 S\_FLIPX** `#define S_FLIPX 0x02U`  
If set the background tile will be flipped horizontally.

**20.67.2.19 S\_FLIPY** `#define S_FLIPY 0x04U`  
If set the background tile will be flipped vertically.

**20.67.2.20 S\_PALETTE** `#define S_PALETTE 0x08U`  
If set the background tile palette.

**20.67.2.21 S\_PRIORITY** `#define S_PRIORITY 0x10U`  
If set the background tile priority.

**20.67.2.22 S\_PAL** `#define S_PAL(`  
                  `n ) ((n) & 0x01U) << 3)`  
Defines how palette number is encoded in OAM. Required for the png2asset tool's metasprite output.

**20.67.2.23 \_\_WRITE\_VDP\_REG\_UNSAFE** `#define __WRITE_VDP_REG_UNSAFE(`  
                  `REG,`  
                  `v ) shadow_##REG=(v),VDP_CMD=(shadow_##REG),VDP_CMD=REG`

**20.67.2.24 \_\_WRITE\_VDP\_REG** `#define __WRITE_VDP_REG(`  
                  `REG,`  
                  `v ) shadow_##REG=(v);__asm__("di");VDP_CMD=(shadow_##REG);VDP_CMD=REG;__asm__↵`  
`("ei")`

**20.67.2.25 \_\_READ\_VDP\_REG** `#define __READ_VDP_REG(`  
                  `REG ) shadow_##REG`

**20.67.2.26 EMPTY\_IFLAG** `#define EMPTY_IFLAG 0x00U`  
Disable calling of interrupt service routines

**20.67.2.27 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](#)



**20.67.2.28 LCD\_IFLAG** `#define LCD_IFLAG 0x02U`  
LCD Interrupt when triggered by the STAT register.

See also

[set\\_interrupts\(\)](#),  
[add\\_LCD](#)

**20.67.2.29 TIM\_IFLAG** `#define TIM_IFLAG 0x04U`  
Does nothing on MSX

**20.67.2.30 SIO\_IFLAG** `#define SIO_IFLAG 0x08U`  
Does nothing on MSX

**20.67.2.31 JOY\_IFLAG** `#define JOY_IFLAG 0x10U`  
Does nothing on MSX

**20.67.2.32 SCREENWIDTH** `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`  
Width of the visible screen in pixels.

**20.67.2.33 SCREENHEIGHT** `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`  
Height of the visible screen in pixels.

**20.67.2.34 MINWNDPOSX** `#define MINWNDPOSX 0x00U`  
The Minimum X position of the Window Layer (Left edge of screen)

See also

[move\\_win\(\)](#)

**20.67.2.35 MINWNDPOSY** `#define MINWNDPOSY 0x00U`  
The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move\\_win\(\)](#)

**20.67.2.36 MAXWNDPOSX** `#define MAXWNDPOSX 0x00U`  
The Maximum X position of the Window Layer (Right edge of screen)

See also

[move\\_win\(\)](#)

**20.67.2.37 MAXWNDPOSY** `#define MAXWNDPOSY 0x00U`  
The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move\\_win\(\)](#)

**20.67.2.38 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](#)

**20.67.2.39 DISPLAY\_OFF** `#define DISPLAY_OFF display_off();`

Turns the display off immediately.

See also

[display\\_off](#), [DISPLAY\\_ON](#)

**20.67.2.40 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](#)

**20.67.2.41 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](#)

**20.67.2.42 SET\_BORDER\_COLOR** `#define SET_BORDER_COLOR(C) __WRITE_VDP_REG(VDP_R7, ((C) | 0xf0u))`

Sets border color

**20.67.2.43 SHOW\_BKG** `#define SHOW_BKG`

Turns on the background layer. Not yet implemented

**20.67.2.44 HIDE\_BKG** `#define HIDE_BKG`

Turns off the background layer. Not yet implemented

**20.67.2.45 SHOW\_WIN** `#define SHOW_WIN`

Turns on the window layer Not yet implemented

**20.67.2.46 HIDE\_WIN** `#define HIDE_WIN`

Turns off the window layer. Not yet implemented

**20.67.2.47 SHOW\_SPRITES** `#define SHOW_SPRITES`

Turns on the sprites layer. Not yet implemented

**20.67.2.48 HIDE\_SPRITES** `#define HIDE_SPRITES`

Turns off the sprites layer. Not yet implemented

**20.67.2.49 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.

**20.67.2.50 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.

**20.67.2.51 DEVICE\_SUPPORTS\_COLOR** `#define DEVICE_SUPPORTS_COLOR (TRUE)`

Macro returns TRUE if device supports color (it always does on MSX)

**20.67.2.52 DIV\_REG** `#define DIV_REG get_r_reg()`

**20.67.2.53 CURRENT\_BANK** `#define CURRENT_BANK _current_bank`

**20.67.2.54 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\(\)](#)

**20.67.2.55 BANKREF** `#define BANKREF( VARNAME )`

#### Value:

```
void __func_ ## VARNAME(void) __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.

**20.67.2.56 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 <a href="#">BANKREF()</a> |
|----------------|----------------------------------------------------------|

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

**See also**

[BANKREF\(\)](#), [BANK\(\)](#)

**20.67.2.57 SWITCH\_ROM1** `#define SWITCH_ROM1 SWITCH\_ROM`

**20.67.2.58 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 |
|----------|-----------------------|

**20.67.2.59 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 |
|----------|------------------------|

**20.67.2.60 ENABLE\_RAM** `#define ENABLE_RAM RAM_CONTROL|=RAMCTL\_RAM`  
Enables RAM

**20.67.2.61 DISABLE\_RAM** `#define DISABLE_RAM RAM_CONTROL&=(~RAMCTL\_RAM)`  
Disables RAM

**20.67.2.62 set\_bkg\_palette\_entry** `#define set_bkg_palette_entry set\_palette\_entry`

**20.67.2.63 set\_sprite\_palette\_entry** `#define set_sprite_palette_entry(  
    palette,  
    entry,  
    rgb_data ) set\_palette\_entry(1,entry,rgb_data)`

**20.67.2.64 set\_bkg\_palette** `#define set_bkg_palette set\_palette`

**20.67.2.65 set\_sprite\_palette** `#define set_sprite_palette(  
    first_palette,  
    nb_palettes,  
    rgb_data ) set\_palette(1,1,rgb_data)`

**20.67.2.66 COMPAT\_PALETTE** `#define COMPAT_PALETTE(  
     C0,  
     C1,  
     C2,  
     C3 ) (((uint16_t) (C3) << 12) | ((uint16_t) (C2) << 8) | ((uint16_t) (C1) << 4) |  
     (uint16_t) (C0))`

**20.67.2.67 set\_bkg\_tiles** `#define set_bkg_tiles set_tile_map`

**20.67.2.68 set\_win\_tiles** `#define set_win_tiles set_tile_map`

**20.67.2.69 fill\_bkg\_rect** `#define fill_bkg_rect fill_rect`

**20.67.2.70 fill\_win\_rect** `#define fill_win_rect fill_rect`

**20.67.2.71 DISABLE\_VBL\_TRANSFER** `#define DISABLE_VBL_TRANSFER _shadow_OAM_base = 0`  
 Disable shadow OAM to VRAM copy on each VBlank

**20.67.2.72 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

**20.67.2.73 MAX\_HARDWARE\_SPRITES** `#define MAX_HARDWARE_SPRITES 32`  
 Amount of hardware sprites in OAM

**20.67.2.74 HARDWARE\_SPRITE\_CAN\_FLIP\_X** `#define HARDWARE_SPRITE_CAN_FLIP_X 0`  
 True if sprite hardware can flip sprites by X (horizontally)

**20.67.2.75 HARDWARE\_SPRITE\_CAN\_FLIP\_Y** `#define HARDWARE_SPRITE_CAN_FLIP_Y 0`  
 True if sprite hardware can flip sprites by Y (vertically)

**20.67.2.76 set\_bkg\_tile\_xy** `#define set_bkg_tile_xy set_tile_xy`

**20.67.2.77 set\_win\_tile\_xy** `#define set_win_tile_xy set_tile_xy`

**20.67.2.78 get\_win\_xy\_addr** `#define get_win_xy_addr get_bkg_xy_addr`

## 20.67.3 Typedef Documentation

**20.67.3.1 int\_handler** `typedef void(* int_handler) (void) NONBANKED`  
 Interrupt handlers

**20.67.3.2 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 <a href="#">set_sprite_tile</a> ) |
| <i>prop</i> | OAM Property Flags (see <a href="#">set_sprite_prop</a> ) |

## 20.67.4 Function Documentation

**20.67.4.1 WRITE\_VDP\_CMD()** `void WRITE_VDP_CMD (`  
`uint16_t cmd )`

**20.67.4.2 WRITE\_VDP\_DATA()** `void WRITE_VDP_DATA (`  
`uint16_t data )`

**20.67.4.3 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](#)

**20.67.4.4 get\_mode()** `uint8_t get_mode (`  
`void )`

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](#)

**20.67.4.5 get\_system()** `uint8_t get_system (`  
`void ) [inline]`

Returns the system gbdk is running on.

**20.67.4.6 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](#)

**20.67.4.7 remove\_VBL()** `void remove_VBL (`  
    `int_handler h )`

Removes the VBL interrupt handler.

**See also**

[add\\_VBL\(\)](#)

**20.67.4.8 remove\_LCD()** `void remove_LCD (`  
    `int_handler h )`

Removes the LCD interrupt handler.

**See also**

[add\\_LCD\(\)](#), [remove\\_VBL\(\)](#)

**20.67.4.9 remove\_TIM()** `void remove_TIM (`  
    `int_handler h )`

**20.67.4.10 remove\_SIO()** `void remove_SIO (`  
    `int_handler h )`

**20.67.4.11 remove\_JOY()** `void remove_JOY (`  
    `int_handler h )`

**20.67.4.12 add\_VBL()** `void add_VBL (`  
    `int_handler h )`

Adds a V-blank interrupt handler.

**20.67.4.13 add\_LCD()** `void add_LCD (`  
    `int_handler h )`

Adds a LCD interrupt handler.

**20.67.4.14 add\_TIM()** `void add_TIM (`  
    `int_handler h )`

Does nothing on MSX

**20.67.4.15 add\_SIO()** `void add_SIO (`  
    `int_handler h )`

Does nothing on MSX

**20.67.4.16 add\_JOY()** `void add_JOY (`  
    `int_handler h )`

Does nothing on MSX

**20.67.4.17 cancel\_pending\_interrupts()** `uint8_t cancel_pending_interrupts (`  
    `void ) [inline]`

Cancel pending interrupts

**20.67.4.18 move\_bkg()** `void move_bkg (`  
    `uint8_t x,`  
    `uint8_t y ) [inline]`

**20.67.4.19 scroll\_bkg()** `void scroll_bkg (`  
    `int8_t x,`  
    `int8_t y ) [inline]`

**20.67.4.20 vsync()** `void vsync (`  
    `void )`

HALTs the CPU and waits for the vertical blank interrupt.

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.

**20.67.4.21 wait\_vbl\_done()** `void wait_vbl_done (`  
    `void )`

Obsolete. This function has been replaced by [vsync\(\)](#), which has identical behavior.

**20.67.4.22 display\_off()** `void display_off (`  
    `void ) [inline]`

Turns the display off.

See also

[DISPLAY\\_ON](#)

**20.67.4.23 refresh\_OAM()** `void refresh_OAM (`  
    `void )`

Copies data from shadow OAM to OAM

**20.67.4.24 get\_r\_reg()** `uint8_t get_r_reg (`  
    `void )`

Return R register for the DIV\_REG emulation

Increments once per CPU instruction (fetches the Z80 CPU R register)

**20.67.4.25 SWITCH\_ROM()** `void SWITCH_ROM (`  
    `uint8_t bank )`

Makes switch the active ROM bank in frame 1

Parameters

|             |                       |
|-------------|-----------------------|
| <i>bank</i> | ROM bank to switch to |
|-------------|-----------------------|



**20.67.4.26 delay()** `void delay (`  
    `uint16_t d )`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

**20.67.4.27 joypad()** `uint8_t joypad (`  
    `void )`

Reads and returns the current state of the joypad.

**20.67.4.28 waitpad()** `uint8_t waitpad (`  
    `uint8_t mask )`

Waits until at least one of the buttons given in mask are pressed.

**20.67.4.29 waitpadup()** `void waitpadup (`  
    `void )`

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.

**20.67.4.30 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 <code>joypads_t</code> 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`

**20.67.4.31 joypad\_ex()** `void joypad_ex (`  
    `joypads_t * joypads )`

Polls all available joypads

#### Parameters

|                |                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>joypads</i> | 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`

**20.67.4.32 enable\_interrupts()** `void enable_interrupts (`  
    `void ) [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](#)

**20.67.4.33 disable\_interrupts()** `void disable_interrupts (`  
`void ) [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](#)

**20.67.4.34 set\_default\_palette()** `void set_default_palette (`  
`void )`

**20.67.4.35 cpu\_fast()** `void cpu_fast (`  
`void ) [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](#)

**20.67.4.36 set\_palette\_entry()** `void set_palette_entry (`  
`uint8_t palette,`  
`uint8_t entry,`  
`uint16_t rgb_data )`

**20.67.4.37 set\_palette()** `void set_palette (`  
`uint8_t first_palette,`  
`uint8_t nb_palettes,`  
`const palette_color_t * rgb_data )`

- 20.67.4.38 set\_native\_tile\_data()** void set\_native\_tile\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src )
- 20.67.4.39 set\_bkg\_4bpp\_data()** void set\_bkg\_4bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]
- 20.67.4.40 set\_sprite\_1bpp\_data()** void set\_sprite\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]
- 20.67.4.41 set\_native\_sprite\_data()** void set\_native\_sprite\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]
- 20.67.4.42 set\_2bpp\_palette()** void set\_2bpp\_palette (  
    uint16\_t palette ) [inline]
- 20.67.4.43 set\_bkg\_data()** void set\_bkg\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]
- 20.67.4.44 set\_sprite\_data()** void set\_sprite\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]
- 20.67.4.45 set\_1bpp\_colors()** void set\_1bpp\_colors (  
    uint8\_t fgcolor,  
    uint8\_t bgcolor ) [inline]
- 20.67.4.46 set\_tile\_1bpp\_data()** void set\_tile\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src,  
    uint16\_t colors )
- 20.67.4.47 set\_bkg\_1bpp\_data()** void set\_bkg\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.67.4.48 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**.

**20.67.4.49 vmemcpy()** `void vmemcpy (`  
    `uint16_t dst,`  
    `const void * src,`  
    `uint16_t size )`

**20.67.4.50 set\_tile\_map()** `void set_tile_map (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t w,`  
    `uint8_t h,`  
    `const uint8_t * tiles )`

**20.67.4.51 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]`

**20.67.4.52 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]`

**20.67.4.53 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 )`

**20.67.4.54 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 )`

**20.67.4.55 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_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](#)

**20.67.4.56 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> ↔<br><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](#)

#### 20.67.4.57 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]
```

#### 20.67.4.58 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]
```

**20.67.4.59 fill\_rect()** `void fill_rect (`  
     `uint8_t x,`  
     `uint8_t y,`  
     `uint8_t w,`  
     `uint8_t h,`  
     `const uint16_t tile )`

**20.67.4.60 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

**20.67.4.61 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<br>In CGB Mode this could be either in VRAM Bank<br>0 or 1, depending on Bit 3 of the OAM Attribute Flag<br>(see <a href="#">set_sprite_prop</a> ) |

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](#)

**20.67.4.62 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

**20.67.4.63 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 - Vertical flip. Dictates which way up the sprite is drawn vertically.  
0: normal  
1: upside down
- Bit 6 - Horizontal flip. Dictates which way up the sprite is drawn horizontally.  
0: normal  
1: back to front
- Bit 5 - Priority flag. When this is set, the sprites appear behind the background and window layer.  
0: infront  
1: behind
- Bit 4 - Unimplemented
- Bit 3 - Unimplemented
- Bit 2 - Unimplemented
- Bit 1 - See bit 0.
- Bit 0 - Bits 0-1 indicate which color palette the sprite should use. Note: only palettes 4 to 7 will be available for NES sprites.

It's recommended to use GBDK constants (eg: `S_FLIPY`) to configure sprite properties as these are crossplatform.

```
// Load palette data into the first palette
set_sprite_palette(4, 1, exampleSprite_palettes)
// Set the OAM value for the sprite
// These flags tell the sprite to use the first sprite palette (palette 4) and to flip the sprite both
//   vertically and horizontally.
set_sprite_prop(0, S_FLIPY | S_FLIPX);
```

See also

[S\\_PALETTE](#), [S\\_FLIPX](#), [S\\_FLIPY](#), [S\\_PRIORITY](#)

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.

It's recommended to use GBDK constants (eg: S\_FLIPY) to configure sprite properties as these are crossplatform.

```
// Load palette data into the first palette
set_sprite_palette(4, 1, exampleSprite_palettes)
// Set the OAM value for the sprite
// These flags tell the sprite to flip both vertically and horizontally.
set_sprite_prop(0, S_FLIPY | S_FLIPX);
```

See also

[S\\_PALETTE](#), [S\\_FLIPX](#), [S\\_FLIPY](#), [S\\_PRIORITY](#)

20.67.4.64 **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

20.67.4.65 **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).<br>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).<br>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.

20.67.4.66 **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 <b>X axis</b><br>Range: -128 - 127 |

Parameters

|          |                                                                               |
|----------|-------------------------------------------------------------------------------|
| <i>y</i> | Number of pixels to move the sprite on the <b>Y axis</b><br>Range: -128 - 127 |
|----------|-------------------------------------------------------------------------------|

See also

[move\\_sprite](#) for more details about the X and Y position

**20.67.4.67 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 |
|-----------|-----------------------------|

**20.67.4.68 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               |

**20.67.4.69 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

**20.67.4.70 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

**20.67.4.71** `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

**20.67.5 Variable Documentation**

**20.67.5.1** `_SYSTEM` `const uint8_t _SYSTEM [extern]`

**20.67.5.2** `c` `void c`

**20.67.5.3** `d` `void d`

**20.67.5.4** `e` `void e`

**20.67.5.5** `iyh` `void iyh`

**20.67.5.6** `iy1` `uint8_t iy1`

Initial value:

```
{
    __asm__("ei")
}
```

**20.67.5.7** `h` `void h`

**20.67.5.8** `l` `void l`

**20.67.5.9** `sys_time` `volatile uint16_t sys_time [extern]`

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)

**20.67.5.10** `_current_bank` `volatile uint8_t _current_bank` [extern]

Tracks current active ROM bank in frame 1

Tracks current active ROM bank

In most cases the `CURRENT_BANK` macro for this variable is recommended for use instead of the variable itself.

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\(\)](#)

**20.67.5.11** `b` `void b`

**20.67.5.12** `_current_2bpp_palette` `uint16_t _current_2bpp_palette` [extern]

**20.67.5.13** `_current_1bpp_colors` `uint16_t _current_1bpp_colors` [extern]

**20.67.5.14** `_map_tile_offset` `uint8_t _map_tile_offset` [extern]

**20.67.5.15** `_submap_tile_offset` `uint8_t _submap_tile_offset` [extern]

**20.67.5.16** `shadow_OAM` `volatile struct OAM_item_t shadow_OAM[]` [extern]

Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

**20.67.5.17** `_shadow_OAM_base` `volatile uint8_t _shadow_OAM_base` [extern]

MSB of shadow\_OAM address is used by OAM copying routine

MSB of shadow\_OAM address is used by OAM DMA copying routine

**20.67.5.18** `_shadow_OAM_OFF` `volatile uint8_t _shadow_OAM_OFF` [extern]

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](#)

## 20.68 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\\_NES](#)
- #define [SYSTEM\\_BITS\\_NTSC](#) 0x00
- #define [SYSTEM\\_BITS\\_PAL](#) 0x40
- #define [SYSTEM\\_BITS\\_DENDY](#) 0x80
- #define [SYSTEM\\_60HZ](#) 0x00
- #define [SYSTEM\\_50HZ](#) 0x01
- #define [RGB\(r, g, b\) RGB\\_TO\\_NES\(\(\(\(r\) >> 6\) | \(\(\(g\) << 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](#) 0x08U
- #define [J\\_DOWN](#) 0x04U
- #define [J\\_LEFT](#) 0x02U
- #define [J\\_RIGHT](#) 0x01U
- #define [J\\_A](#) 0x80U
- #define [J\\_B](#) 0x40U
- #define [J\\_SELECT](#) 0x20U
- #define [J\\_START](#) 0x10U
- #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 S\_PAL(n) n
- #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 MAX\_LCD\_ISR\_CALLS 4
- #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\_DUMMY(b)
- #define SWITCH\_ROM\_UNROM(b) \_switch\_prg0(b)
- #define SWITCH\_ROM SWITCH\_ROM\_UNROM
- #define SWITCH\_RAM(b) 0
- #define ENABLE\_RAM
- #define DISABLE\_RAM
- #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 SET\_BORDER\_COLOR(C)
- #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\_attribute\_xy set\_bkg\_attribute\_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 HARDWARE\_SPRITE\_CAN\_FLIP\_X 1
- #define HARDWARE\_SPRITE\_CAN\_FLIP\_Y 1
- #define fill\_rect fill\_bkg\_rect

## Typedefs

- typedef uint8\_t palette\_color\_t
- typedef void(\* int\_handler) (void) NONBANKED
- typedef struct OAM\_item\_t OAM\_item\_t

## Functions

- void [set\\_bkg\\_palette](#) (uint8\_t first\_palette, uint8\_t nb\_palettes, const [palette\\_color\\_t](#) \*rgb\_data) NO\_↔OVERLAY\_LOCALS
- void [set\\_sprite\\_palette](#) (uint8\_t first\_palette, uint8\_t nb\_palettes, const [palette\\_color\\_t](#) \*rgb\_data) NO\_↔OVERLAY\_LOCALS
- void [set\\_bkg\\_palette\\_entry](#) (uint8\_t palette, uint8\_t entry, [palette\\_color\\_t](#) rgb\_data) NO\_OVERLAY\_LOCALS
- void [set\\_sprite\\_palette\\_entry](#) (uint8\_t palette, uint8\_t entry, [palette\\_color\\_t](#) rgb\_data) NO\_OVERLAY\_↔LOCALS
- void [remove\\_VBL](#) (int\_handler h) NO\_OVERLAY\_LOCALS
- void [remove\\_LCD](#) (int\_handler h) NO\_OVERLAY\_LOCALS
- void [add\\_VBL](#) (int\_handler h) NO\_OVERLAY\_LOCALS
- void [add\\_LCD](#) (int\_handler h) NO\_OVERLAY\_LOCALS
- void [mode](#) (uint8\_t m) NO\_OVERLAY\_LOCALS
- [uint8\\_t get\\_mode](#) (void) NO\_OVERLAY\_LOCALS
- [uint8\\_t get\\_system](#) (void)
- void [delay](#) (uint16\_t d) NO\_OVERLAY\_LOCALS
- [uint8\\_t joypad](#) (void) NO\_OVERLAY\_LOCALS
- [uint8\\_t waitpad](#) (uint8\_t mask) NO\_OVERLAY\_LOCALS
- void [waitpadup](#) (void) NO\_OVERLAY\_LOCALS
- [uint8\\_t joypad\\_init](#) (uint8\_t npads, [joypads\\_t](#) \*joypads) NO\_OVERLAY\_LOCALS
- void [joypad\\_ex](#) ([joypads\\_t](#) \*joypads) NO\_OVERLAY\_LOCALS
- void [enable\\_interrupts](#) (void)
- void [disable\\_interrupts](#) (void)
- void [vsync](#) (void) NO\_OVERLAY\_LOCALS
- void [wait\\_vbl\\_done](#) (void) NO\_OVERLAY\_LOCALS
- void [display\\_on](#) (void) NO\_OVERLAY\_LOCALS
- void [display\\_off](#) (void) NO\_OVERLAY\_LOCALS
- void [refresh\\_OAM](#) (void) NO\_OVERLAY\_LOCALS
- void [set\\_vram\\_byte](#) (uint8\_t \*addr, uint8\_t v) NO\_OVERLAY\_LOCALS
- [uint8\\_t \\* get\\_bkg\\_xy\\_addr](#) (uint8\_t x, uint8\_t y) NO\_OVERLAY\_LOCALS
- void [set\\_2bpp\\_palette](#) (uint16\_t palette)
- void [set\\_1bpp\\_colors\\_ex](#) (uint8\_t fgcolor, uint8\_t bgcolor, uint8\_t mode) NO\_OVERLAY\_LOCALS
- 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) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_1bpp\\_data](#) (uint8\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_tiles](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*tiles) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_attributes\\_nes16x16](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*attributes) NO\_↔OVERLAY\_LOCALS
- void [set\\_bkg\\_attributes](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*attributes)
- void [set\\_bkg\\_submap\\_attributes\\_nes16x16](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*map, uint8\_t map\_w) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_submap\\_attributes](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*attributes, uint8\_t map\_w)
- 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) NO\_↔OVERLAY\_LOCALS
- 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) NO\_OVERLAY\_LOCALS
- [uint8\\_t \\* set\\_bkg\\_tile\\_xy](#) (uint8\_t x, uint8\_t y, uint8\_t t) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_attribute\\_xy\\_nes16x16](#) (uint8\_t x, uint8\_t y, uint8\_t a) NO\_OVERLAY\_LOCALS
- void [set\\_bkg\\_attribute\\_xy](#) (uint8\_t x, uint8\_t y, uint8\_t a)
- [uint8\\_t get\\_bkg\\_tile\\_xy](#) (uint8\_t x, uint8\_t y) NO\_OVERLAY\_LOCALS
- 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) NO\_OVERLAY\_LOCALS
- void `set_sprite_1bpp_data` (uint8\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data) NO\_OVERLAY\_LOCALS
- void `SET_SHADOW_OAM_ADDRESS` (void \*address)
- void `set_sprite_tile` (uint8\_t nb, uint8\_t tile) NO\_OVERLAY\_LOCALS
- uint8\_t `get_sprite_tile` (uint8\_t nb) NO\_OVERLAY\_LOCALS
- void `set_sprite_prop` (uint8\_t nb, uint8\_t prop) NO\_OVERLAY\_LOCALS
- uint8\_t `get_sprite_prop` (uint8\_t nb) NO\_OVERLAY\_LOCALS
- void `move_sprite` (uint8\_t nb, uint8\_t x, uint8\_t y) NO\_OVERLAY\_LOCALS
- void `scroll_sprite` (uint8\_t nb, int8\_t x, int8\_t y) NO\_OVERLAY\_LOCALS
- void `hide_sprite` (uint8\_t nb) NO\_OVERLAY\_LOCALS
- void `set_data` (uint8\_t \*vram\_addr, const uint8\_t \*data, uint16\_t len) NO\_OVERLAY\_LOCALS
- void `set_tiles` (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, uint8\_t \*vram\_addr, const uint8\_t \*tiles) NO\_OVERLAY\_LOCALS
- void `set_tile_data` (uint16\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data)
- void `set_bkg_native_data` (uint8\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data) NO\_OVERLAY\_LOCALS
- void `set_sprite_native_data` (uint8\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data) NO\_OVERLAY\_LOCALS
- void `set_native_tile_data` (uint16\_t first\_tile, uint8\_t nb\_tiles, const uint8\_t \*data)
- void `init_bkg` (uint8\_t c) NO\_OVERLAY\_LOCALS
- void `vmemset` (void \*s, uint8\_t c, size\_t n) NO\_OVERLAY\_LOCALS
- void `fill_bkg_rect` (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, uint8\_t tile) NO\_OVERLAY\_LOCALS
- void `flush_shadow_attributes` (void) NO\_OVERLAY\_LOCALS
- uint8\_t `_switch_prg0` (uint8\_t bank) NO\_OVERLAY\_LOCALS

## Variables

- const uint8\_t `_SYSTEM`
- 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`

## 20.68.1 Detailed Description

NES specific functions.

## 20.68.2 Macro Definition Documentation

**20.68.2.1 NINTENDO\_NES** `#define NINTENDO_NES`

**20.68.2.2 SYSTEM\_BITS\_NTSC** `#define SYSTEM_BITS_NTSC 0x00`

**20.68.2.3 SYSTEM\_BITS\_PAL** `#define SYSTEM_BITS_PAL 0x40`

**20.68.2.4 SYSTEM\_BITS\_DENDY** `#define SYSTEM_BITS_DENDY 0x80`

**20.68.2.5 SYSTEM\_60HZ** `#define SYSTEM_60HZ 0x00`



**20.68.2.6 SYSTEM\_50HZ** `#define SYSTEM_50HZ 0x01`

**20.68.2.7 RGB** `#define RGB(  
 r,  
 g,  
 b ) RGB_TO_NES(((r) | ((g) << 2) | ((b) << 4)))`

**20.68.2.8 RGB8** `#define RGB8(  
 r,  
 g,  
 b ) RGB_TO_NES((((r) >> 6) | (((g) >> 6) << 2) | (((b) >> 6) << 4)))`

**20.68.2.9 RGBHTML** `#define RGBHTML(  
 RGB24bit ) RGB_TO_NES((((RGB24bit) >> 22) | (((RGB24bit) & 0xFFFF) >> 14) <<  
 2) | (((RGB24bit) & 0xFF) >> 6) << 4)))`

**20.68.2.10 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](https://www.nesdev.org/wiki/PPU_palettes#RGBI)

**20.68.2.11 RGB\_DARKRED** `#define RGB_DARKRED 0x06`

**20.68.2.12 RGB\_GREEN** `#define RGB_GREEN 0x2A`

**20.68.2.13 RGB\_DARKGREEN** `#define RGB_DARKGREEN 0x1A`

**20.68.2.14 RGB\_BLUE** `#define RGB_BLUE 0x12`

**20.68.2.15 RGB\_DARKBLUE** `#define RGB_DARKBLUE 0x02`

**20.68.2.16 RGB\_YELLOW** `#define RGB_YELLOW 0x28`

**20.68.2.17 RGB\_DARKYELLOW** `#define RGB_DARKYELLOW 0x18`

**20.68.2.18 RGB\_CYAN** `#define RGB_CYAN 0x2C`

**20.68.2.19 RGB\_AQUA** `#define RGB_AQUA 0x1C`

**20.68.2.20 RGB\_PINK** `#define RGB_PINK 0x24`

**20.68.2.21 RGB\_PURPLE** `#define RGB_PURPLE 0x14`

**20.68.2.22 RGB\_BLACK** `#define RGB_BLACK 0x0F`

**20.68.2.23 RGB\_DARKGRAY** `#define RGB_DARKGRAY 0x00`

**20.68.2.24 RGB\_LIGHTGRAY** `#define RGB_LIGHTGRAY 0x10`

**20.68.2.25 RGB\_WHITE** `#define RGB_WHITE 0x30`

**20.68.2.26 J\_UP** `#define J_UP 0x08U`

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](#)

**20.68.2.27 J\_DOWN** `#define J_DOWN 0x04U`

**20.68.2.28 J\_LEFT** `#define J_LEFT 0x02U`

**20.68.2.29 J\_RIGHT** `#define J_RIGHT 0x01U`

**20.68.2.30 J\_A** `#define J_A 0x80U`

**20.68.2.31 J\_B** `#define J_B 0x40U`

**20.68.2.32 J\_SELECT** `#define J_SELECT 0x20U`

**20.68.2.33 J\_START** `#define J_START 0x10U`

**20.68.2.34 M\_DRAWING** `#define M_DRAWING 0x01U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

**20.68.2.35 M\_TEXT\_OUT** `#define M_TEXT_OUT 0x02U`

**20.68.2.36 M\_TEXT\_INOUT** `#define M_TEXT_INOUT 0x03U`

**20.68.2.37 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\(\)](#)

**20.68.2.38 M\_NO\_INTERP** `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

**20.68.2.39 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\(\)](#).

**20.68.2.40 S\_FLIPX** `#define S_FLIPX 0x40U`

If set the sprite will be flipped horizontally.

See also

[set\\_sprite\\_prop\(\)](#)

**20.68.2.41 S\_FLIPY** `#define S_FLIPY 0x80U`

If set the sprite will be flipped vertically.

See also

[set\\_sprite\\_prop\(\)](#)

**20.68.2.42 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\(\)](#)

**20.68.2.43 S\_PAL** `#define S_PAL(  
n ) n`

Defines how palette number is encoded in OAM. Required for the png2asset tool's metasprite output.

**20.68.2.44 DMG\_BLACK** `#define DMG_BLACK 0x03`

**20.68.2.45 DMG\_DARK\_GRAY** `#define DMG_DARK_GRAY 0x02`

**20.68.2.46 DMG\_LITE\_GRAY** `#define DMG_LITE_GRAY 0x01`

**20.68.2.47 DMG\_WHITE** `#define DMG_WHITE 0x00`

**20.68.2.48 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](#)

**20.68.2.49 SCREENWIDTH** `#define SCREENWIDTH DEVICE\_SCREEN\_PX\_WIDTH`

Width of the visible screen in pixels.

**20.68.2.50 SCREENHEIGHT** `#define SCREENHEIGHT DEVICE\_SCREEN\_PX\_HEIGHT`

Height of the visible screen in pixels.

**20.68.2.51 MAX\_LCD\_ISR\_CALLS** `#define MAX_LCD_ISR_CALLS 4`

The maximum number of times the LCD handler will be called per frame.

**20.68.2.52 CURRENT\_BANK** `#define CURRENT_BANK \_\_current\_bank`

**20.68.2.53 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\(\)](#)

**20.68.2.54 BANKREF** `#define BANKREF(  
VARNAME )`

**Value:**

```
void __func_ ## VARNAME(void) __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.

**20.68.2.55 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 <a href="#">BANKREF()</a> |
|----------------|----------------------------------------------------------|

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

**20.68.2.56 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 |
|----------|-----------------------|

**20.68.2.57 SWITCH\_ROM\_UNROM** `#define SWITCH_ROM_UNROM(  
b ) _switch_prg0(b)`

Macro for simple UNROM-like switching (write bank# to single 8-bit register)

**Parameters**

|          |                       |
|----------|-----------------------|
| <i>b</i> | ROM bank to switch to |
|----------|-----------------------|

**20.68.2.58 SWITCH\_ROM** `#define SWITCH_ROM SWITCH_ROM_UNROM`

Makes default mapper switch the active ROM bank

**Parameters**

|          |                                 |
|----------|---------------------------------|
| <i>b</i> | ROM bank to switch to (max 255) |
|----------|---------------------------------|

**See also**

[SWITCH\\_ROM\\_UNROM](#)

**20.68.2.59 SWITCH\_RAM** `#define SWITCH_RAM(  
    b ) 0`

No-op at the moment. Placeholder for future mappers / test compatibility.

**Parameters**

|          |                        |
|----------|------------------------|
| <i>b</i> | SRAM bank to switch to |
|----------|------------------------|

**20.68.2.60 ENABLE\_RAM** `#define ENABLE_RAM`

No-op at the moment. Placeholder for future mappers / test compatibility.

**20.68.2.61 DISABLE\_RAM** `#define DISABLE_RAM`

No-op at the moment. Placeholder for future mappers / test compatibility.

**20.68.2.62 DISPLAY\_ON** `#define DISPLAY_ON display_on();`

Turns the display back on.

**See also**

[display\\_off](#), [DISPLAY\\_OFF](#)

**20.68.2.63 DISPLAY\_OFF** `#define DISPLAY_OFF display_off();`

Turns the display off immediately.

**See also**

[display\\_off](#), [DISPLAY\\_ON](#)

**20.68.2.64 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](#)

**20.68.2.65 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](#)

**20.68.2.66 SET\_BORDER\_COLOR** `#define SET_BORDER_COLOR( C )`

Does nothing for NES not implemented yet

**20.68.2.67 SHOW\_BKG** `#define SHOW_BKG shadow_PPUMASK |= PPUMASK_SHOW_BG;`

Turns on the background layer. Sets bit 0 of the LCDC register to 1.

**20.68.2.68 HIDE\_BKG** `#define HIDE_BKG shadow_PPUMASK &= ~PPUMASK_SHOW_BG;`

Turns off the background layer. Sets bit 0 of the LCDC register to 0.

**20.68.2.69 SHOW\_SPRITES** `#define SHOW_SPRITES shadow_PPUMASK |= PPUMASK_SHOW_SPR;`

Turns on the sprites layer. Sets bit 1 of the LCDC register to 1.

**20.68.2.70 HIDE\_SPRITES** `#define HIDE_SPRITES shadow_PPUMASK &= ~PPUMASK_SHOW_SPR;`

Turns off the sprites layer. Clears bit 1 of the LCDC register to 0.

**20.68.2.71 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.

**20.68.2.72 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.

**20.68.2.73 COMPAT\_PALETTE** `#define COMPAT_PALETTE( C0, C1, C2, C3 ) ((uint8_t)((C3) << 6) | ((C2) << 4) | ((C1) << 2) | (C0))`

**20.68.2.74 set\_bkg\_2bpp\_data** `#define set_bkg_2bpp_data set_bkg_data`

**20.68.2.75 set\_tile\_map** `#define set_tile_map set_bkg_tiles`

**20.68.2.76 set\_tile\_submap** `#define set_tile_submap set_bkg_submap`

**20.68.2.77 set\_tile\_xy** `#define set_tile_xy set_bkg_tile_xy`

**20.68.2.78 set\_attribute\_xy** `#define set_attribute_xy set_bkg_attribute_xy`

**20.68.2.79 set\_sprite\_2bpp\_data** `#define set_sprite_2bpp_data set_sprite_data`

**20.68.2.80 DISABLE\_OAM\_DMA** `#define DISABLE_OAM_DMA \_shadow\_OAM\_base = 0`

**20.68.2.81 DISABLE\_VBL\_TRANSFER** `#define DISABLE_VBL_TRANSFER DISABLE\_OAM\_DMA`  
Disable OAM DMA copy each VBlank

**20.68.2.82 ENABLE\_OAM\_DMA** `#define ENABLE_OAM_DMA \_shadow\_OAM\_base = (uint8_t)((uint16_t)&shadow\_OAM >> 8)`

**20.68.2.83 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

**20.68.2.84 MAX\_HARDWARE\_SPRITES** `#define MAX_HARDWARE_SPRITES 64`  
Amount of hardware sprites in OAM

**20.68.2.85 HARDWARE\_SPRITE\_CAN\_FLIP\_X** `#define HARDWARE_SPRITE_CAN_FLIP_X 1`  
True if sprite hardware can flip sprites by X (horizontally)

**20.68.2.86 HARDWARE\_SPRITE\_CAN\_FLIP\_Y** `#define HARDWARE_SPRITE_CAN_FLIP_Y 1`  
True if sprite hardware can flip sprites by Y (vertically)

**20.68.2.87 fill\_rect** `#define fill_rect fill\_bkg\_rect`

## 20.68.3 Typedef Documentation

**20.68.3.1 palette\_color\_t** `typedef uint8\_t palette\_color\_t`

**20.68.3.2 int\_handler** `typedef void(* int_handler) (void) NONBANKED`  
Interrupt handlers

**20.68.3.3 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 <a href="#">set_sprite_tile</a> ) |
| <i>prop</i> | OAM Property Flags (see <a href="#">set_sprite_prop</a> ) |

## 20.68.4 Function Documentation

**20.68.4.1 set\_bkg\_palette()** `void set_bkg_palette (   
    uint8\_t first_palette,   
    uint8\_t nb_palettes,   
    const palette\_color\_t * rgb_data )`



**20.68.4.2 set\_sprite\_palette()** void set\_sprite\_palette (   
     uint8\_t first\_palette,   
     uint8\_t nb\_palettes,   
     const palette\_color\_t \* rgb\_data )

**20.68.4.3 set\_bkg\_palette\_entry()** void set\_bkg\_palette\_entry (   
     uint8\_t palette,   
     uint8\_t entry,   
     palette\_color\_t rgb\_data )

**20.68.4.4 set\_sprite\_palette\_entry()** void set\_sprite\_palette\_entry (   
     uint8\_t palette,   
     uint8\_t entry,   
     palette\_color\_t rgb\_data )

**20.68.4.5 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\(\)](#)

**20.68.4.6 remove\_LCD()** void remove\_LCD (   
     int\_handler h )

Removes the LCD interrupt handler.

See also

[add\\_LCD\(\)](#), [remove\\_VBL\(\)](#)

**20.68.4.7 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. |
|----------|---------------------------------------------------------------|

Only a single handler is currently supported for NES.

**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.

On the current NES implementation, this handler is actually faked, and called before vblank occurs, by [vsync\(\)](#). Writes to PPU registers should be done to the shadow\_ versions, so they are updated by the default VBL handler only when vblank actually occurs.

**See also**

[ISR\\_VECTOR\(\)](#)

Adds a V-blank interrupt handler.

**20.68.4.8 add\_LCD()** `void add_LCD (`  
`int_handler h )`

Adds a LCD interrupt handler.

Called when the scanline matches the `_lcd_scanline` variables.

Only a single handler is currently supported for NES.

The use-case 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 scrolling registers 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.

**Note**

On the current NES implementation, this handler is actually faked, and called by the default VBL handler after a manual delay loop. Only one such faked "interrupt" is possible per frame. This means the CPU cycles wasted in the delay loop increase with higher values of `_lcd_scanline`. In practice, it makes this functionality mostly suited for a top status bar.

**See also**

[add\\_VBL](#), [nowait\\_int\\_handler](#), [ISR\\_VECTOR\(\)](#)

Adds a LCD interrupt handler.

**20.68.4.9 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](#)

**20.68.4.10 get\_mode()** `uint8_t get_mode (`  
`void )`

Returns the current mode

**See also**

[M\\_DRAWING](#), [M\\_TEXT\\_OUT](#), [M\\_TEXT\\_INOUT](#), [M\\_NO\\_SCROLL](#), [M\\_NO\\_INTERP](#)

**20.68.4.11 get\_system()** `uint8_t get_system (`  
`void ) [inline]`

Returns the system gbdk is running on.

**20.68.4.12 delay()** `void delay (`  
`uint16_t d )`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

**20.68.4.13 joypad()** `uint8_t joypad (`  
`void )`

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.

**20.68.4.14 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.

**20.68.4.15 waitpadup()** `void waitpadup (`  
`void )`

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.

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.

**20.68.4.16** `joypad_init()` `uint8_t joypad_init (`  
     `uint8_t npads,`  
     `joypads_t * joypads )`

Initializes `joypads_t` structure for polling multiple joypads

#### Parameters

|                      |                                                               |
|----------------------|---------------------------------------------------------------|
| <code>npads</code>   | number of joypads requested (1, 2 or 4)                       |
| <code>joypads</code> | pointer to <code>joypads_t</code> 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`

**20.68.4.17** `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

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <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`

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`

**20.68.4.18** `enable_interrupts()` `void enable_interrupts (`  
     `void ) [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](#)

**20.68.4.19 disable\_interrupts()** `void disable_interrupts (`  
`void ) [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](#)

**20.68.4.20 vsync()** `void vsync (`  
`void )`

Waits for the vertical blank interrupt.

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 and then returns when all registered VBL ISRs have completed.

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.

HALTs the CPU and waits for the vertical blank interrupt.

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.

**20.68.4.21 wait\_vbl\_done()** `void wait_vbl_done (`  
`void )`

Obsolete. This function has been replaced by [vsync\(\)](#), which has identical behavior.

**20.68.4.22 display\_on()** `void display_on (`  
`void )`

Turns the display on.

**See also**

[DISPLAY\\_ON](#)

**20.68.4.23 display\_off()** `void display_off (`  
    `void ) [inline]`

Turns the display off immediately.

See also

[DISPLAY\\_ON](#)

Turns the display off.

Waits until the VBL before turning the display off.

See also

[DISPLAY\\_ON](#)

Turns the display off.

See also

[DISPLAY\\_ON](#)

**20.68.4.24 refresh\_OAM()** `void refresh_OAM (`  
    `void )`

Copies data from shadow OAM to OAM

**20.68.4.25 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               |

**20.68.4.26 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

**20.68.4.27 set\_2bpp\_palette()** `void set_2bpp_palette (`  
    `uint16_t palette ) [inline]`

Sets palette for 2bpp color translation for GG/SMS, does nothing on GB

**20.68.4.28 set\_1bpp\_colors\_ex()** `void set_1bpp_colors_ex (`  
    `uint8_t fgcolor,`  
    `uint8_t bgcolor,`  
    `uint8_t mode )`

**20.68.4.29 set\_1bpp\_colors()** `void set_1bpp_colors (`  
    `uint8_t fgcolor,`  
    `uint8_t bgcolor ) [inline]`

**20.68.4.30 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 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](#)

**20.68.4.31 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](#)

**20.68.4.32 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](#)

**20.68.4.33** `set_bkg_attributes_nes16x16()` `void set_bkg_attributes_nes16x16 (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t w,`  
`uint8_t h,`  
`const uint8_t * attributes )`

Sets a rectangular region of Background Tile Map Attributes.

## Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <i>x</i>          | X Start position in Background Map tile coordinates. Range 0 - 15 |
| <i>y</i>          | Y Start position in Background Map tile coordinates. Range 0 - 14 |
| <i>w</i>          | Width of area to set in tiles. Range 1 - 16                       |
| <i>h</i>          | Height of area to set in tiles. Range 1 - 15                      |
| <i>attributes</i> | Pointer to source tile map attribute 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.

NES 16x16 Tile Attributes are tightly packed into 4 attributes per byte, with each 16x16 area of a 32x32 pixel block using the bits as follows: D1-D0: Top-left 16x16 pixels D3-D2: Top-right 16x16 pixels D5-D4: Bottom-left 16x16 pixels D7-D6: Bottom-right 16x16 pixels

[https://www.nesdev.org/wiki/PPU\\_attribute\\_tables](https://www.nesdev.org/wiki/PPU_attribute_tables)

See also

[SHOW\\_BKG](#)

[set\\_bkg\\_data](#), [set\\_bkg\\_submap\\_attributes](#), [set\\_win\\_tiles](#), [set\\_tiles](#)

**20.68.4.34 set\_bkg\_attributes()** `void set_bkg_attributes (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t w,`  
`uint8_t h,`  
`const uint8_t * attributes ) [inline]`

Sets a rectangular region of Background Tile Map Attributes.

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\\_attributes\(\)](#) 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 attribute entry.

Writes that exceed coordinate 31 on the x or y axis will wrap around to the Left and Top edges.

Please note that this is just a wrapper function for [set\\_bkg\\_attributes\\_nes16x16\(\)](#) and divides the coordinates and dimensions by 2 to achieve this. It is intended to make code more portable by using the same coordinate system that systems with the much more common 8x8 attribute resolution would use.

See also

[SHOW\\_BKG](#)

[set\\_bkg\\_data](#), [set\\_bkg\\_submap\\_attributes](#), [set\\_win\\_tiles](#), [set\\_tiles](#)

**20.68.4.35 set\_bkg\_submap\_attributes\_nes16x16()** `void set_bkg_submap_attributes_nes16x16 (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t w,`  
`uint8_t h,`  
`const uint8_t * map,`  
`uint8_t map_w )`

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 30 tiles / 16x15 attributes.

Parameters

|              |                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------|
| <i>x</i>     | X Start position in both the Source Attribute Map and hardware Background Map attribute coordinates. Range 0 - 255 |
| <i>y</i>     | Y Start position in both the Source Attribute Map and hardware Background Map attribute coordinates. Range 0 - 255 |
| <i>w</i>     | Width of area to set in Attributes. Range 1 - 127                                                                  |
| <i>h</i>     | Height of area to set in Attributes. Range 1 - 127                                                                 |
| <i>map</i>   | Pointer to source tile map data                                                                                    |
| <i>map_w</i> | Width of source tile map in tiles. Range 1 - 127                                                                   |

Entries are copied from **map** to the Background Attribute Map starting at **x**, **y** writing across for **w** tiles and down for **h** attributes, using **map\_w** as the rowstride for the source attribute map.

The **x** and **y** parameters are in Source Attribute Map Attribute 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 0-15 (they are bit-masked:  $x \& 0xF$  and  $y \& 0xF$ ). 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 Attribute 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 attribute map entry.

Writes that exceed coordinate 15/14 on the x / 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](#)

**20.68.4.36 set\_bkg\_submap\_attributes()** `void set_bkg_submap_attributes (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t w,`  
`uint8_t h,`  
`const uint8_t * attributes,`  
`uint8_t map_w ) [inline]`

Sets a rectangular area of the Background Tile Map attributes using a sub-region from a source tile map. Useful for scrolling implementations of maps larger than 32 x 30 tiles.

Please note that this is just a wrapper function for [set\\_bkg\\_submap\\_attributes\\_nes16x16\(\)](#) and divides the coordinates and dimensions by 2 to achieve this. It is intended to make code more portable by using the same coordinate system that systems with the much more common 8x8 attribute resolution would use.

See also

[SHOW\\_BKG](#)

[set\\_bkg\\_data](#), [set\\_bkg\\_tiles](#), [set\\_win\\_submap](#), [set\\_tiles](#)

**20.68.4.37 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

**20.68.4.38 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> ↔<br><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](#)

**20.68.4.39 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

**20.68.4.40 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                   |

## Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

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](#)

**20.68.4.41** `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

**20.68.4.42** `set_bkg_attribute_xy_nes16x16()` `void set_bkg_attribute_xy_nes16x16 (`  
`uint8_t x,`  
`uint8_t y,`  
`uint8_t a )`

Set single attribute data a on background layer at x,y

## Parameters

|          |                 |
|----------|-----------------|
| <i>x</i> | X-coordinate    |
| <i>y</i> | Y-coordinate    |
| <i>a</i> | tile attributes |

**20.68.4.43** `set_bkg_attribute_xy()` `void set_bkg_attribute_xy (`

```
uint8_t x,  
uint8_t y,  
uint8_t a ) [inline]
```

Set single attribute data *a* on background layer at *x*,*y*

Please note that this is just a wrapper function for [set\\_bkg\\_submap\\_attributes\\_nes16x16\(\)](#) and divides the coordinates and dimensions by 2 to achieve this. It is intended to make code more portable by using the same coordinate system that systems with the much more common 8x8 attribute resolution would use.

#### Parameters

|          |                 |
|----------|-----------------|
| <i>x</i> | X-coordinate    |
| <i>y</i> | Y-coordinate    |
| <i>a</i> | tile attributes |

**20.68.4.44** [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

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

#### Note

In general **avoid reading from VRAM** since that memory is not accessible at all times. It is also not supported by GBDK on the NES platform. See [coding guidelines](#) for more details.

**20.68.4.45** [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](#)

**20.68.4.46 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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i> | Number of pixels to move the Background on the <b>Y axis</b><br>Range: -128 - 127 |

See also

[move\\_bkg](#)

**20.68.4.47 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 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 tile patterns are written to.

- VBK\_REG = [VBK\\_BANK\\_0](#) indicates the first bank



- VBK\_REG = [VBK\\_BANK\\_1](#) indicates the second

**20.68.4.48 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](#)

**20.68.4.49 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

**20.68.4.50 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](#)

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<br>In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the OAM Attribute Flag (see <a href="#">set_sprite_prop</a> ) |

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](#)

**20.68.4.51** `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

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

**20.68.4.52** `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 - Vertical flip. Dictates which way up the sprite is drawn vertically.  
0: normal  
1: upside down
- Bit 6 - Horizontal flip. Dictates which way up the sprite is drawn horizontally.  
0: normal  
1: back to front
- Bit 5 - Priority flag. When this is set, the sprites appear behind the background and window layer.  
0: infront  
1: behind
- Bit 4 - Unimplemented
- Bit 3 - Unimplemented
- Bit 2 - Unimplemented
- Bit 1 - See bit 0.
- Bit 0 - Bits 0-1 indicate which color palette the sprite should use. Note: only palettes 4 to 7 will be available for NES sprites.

It's recommended to use GBDK constants (eg: `S_FLIPY`) to configure sprite properties as these are crossplatform.

```
// Load palette data into the first palette
set_sprite_palette(4, 1, exampleSprite_palettes)
// Set the OAM value for the sprite
// These flags tell the sprite to use the first sprite palette (palette 4) and to flip the sprite both
//   vertically and horizontally.
set_sprite_prop(0, S_FLIPY | S_FLIPX);
```

See also

[S\\_PALETTE](#), [S\\_FLIPX](#), [S\\_FLIPY](#), [S\\_PRIORITY](#)

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.

It's recommended to use GBDK constants (eg: S\_FLIPY) to configure sprite properties as these are crossplatform.

```
// Load palette data into the first palette
set_sprite_palette(4, 1, exampleSprite_palettes)
// Set the OAM value for the sprite
// These flags tell the sprite to flip both vertically and horizontally.
set_sprite_prop(0, S_FLIPY | S_FLIPX);
```

See also

[S\\_PALETTE](#), [S\\_FLIPX](#), [S\\_FLIPY](#), [S\\_PRIORITY](#)

Function has no affect on sms.

This function is only here to enable game portability

**20.68.4.53 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

**20.68.4.54 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).<br>An offscreen value ( $Y \geq 240$ ) hides the sprite. |

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

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).<br>An offscreen value ( $X=0$ or $X \geq 168$ ) 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).<br>An offscreen value (for example, $Y=0$ or $Y \geq 160$ ) hides the sprite.                                                                                                               |

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

**20.68.4.55 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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i>  | Number of pixels to move the sprite on the <b>Y axis</b><br>Range: -128 - 127 |

#### See also

[move\\_sprite](#) for more details about the X and Y position

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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i>  | Number of pixels to move the sprite on the <b>Y axis</b><br>Range: -128 - 127 |

#### See also

[move\\_sprite](#) for more details about the X and Y position

**20.68.4.56 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 |
|-----------|-----------------------------|

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](#)

Hides sprite number **nb** by moving it to zero position by Y.

#### Parameters

|           |                             |
|-----------|-----------------------------|
| <i>nb</i> | Sprite number, range 0 - 39 |
|-----------|-----------------------------|

**20.68.4.57 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](#)

**20.68.4.58 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](#)

**20.68.4.59 set\_tile\_data()** void set\_tile\_data (   
     uint16\_t first\_tile,   
     uint8\_t nb\_tiles,   
     const uint8\_t \* data ) [inline]

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](#)

**20.68.4.60 set\_bkg\_native\_data()** void set\_bkg\_native\_data (   
     uint8\_t first\_tile,   
     uint8\_t nb\_tiles,   
     const uint8\_t \* data ) [inline]

Sets VRAM Tile Pattern data for the Background in the native format

## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.

See also

[set\\_tile\\_data](#)

Sets VRAM Tile Pattern data for the Background / Window in the native format

## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.

GBC only: [VBK\\_REG](#) determines which bank of 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](#)

**20.68.4.61 set\_sprite\_native\_data()** `void set_sprite_native_data (`  
`uint8_t first_tile,`  
`uint8_t nb_tiles,`  
`const uint8_t * data ) [inline]`

Sets VRAM Tile Pattern data for Sprites in the native format

## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.

Sets VRAM Tile Pattern data for Sprites in the native format

## 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 source tile data      |

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**.

GBC only: [VBK\\_REG](#) determines which bank of tile patterns are written to.

- `VBK_REG = VBK_BANK_0` indicates the first bank



- VBK\_REG = [VBK\\_BANK\\_1](#) indicates the second

**20.68.4.62 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

**20.68.4.63 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  
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

**20.68.4.64 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

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

**20.68.4.65 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                                                        |

**20.68.4.66 flush\_shadow\_attributes()** `void flush_shadow_attributes (`  
     `void )`

"Flushes" the updates to the shadow attributes so they are written to the transfer buffer, and then written to PPU memory on next vblank.

This function must be called to see visible changes to attributes on the NES target. But it will automatically be called by `vsync()`, so the use-cases for calling it manually are rare in practice.

**20.68.4.67 \_switch\_prg0()** `uint8_t _switch_prg0 (`  
     `uint8_t bank )`

## 20.68.5 Variable Documentation

**20.68.5.1 \_SYSTEM** `const uint8_t _SYSTEM [extern]`

**20.68.5.2 sys\_time** `volatile uint16_t sys_time [extern]`

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)

**20.68.5.3** `_current_bank` `volatile uint8_t _current_bank` [extern]

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()`

Tracks current active ROM bank

In most cases the `CURRENT_BANK` macro for this variable is recommended for use instead of the variable itself.

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()`

**20.68.5.4** `_current_1bpp_colors` `uint16_t _current_1bpp_colors` [extern]**20.68.5.5** `_map_tile_offset` `uint8_t _map_tile_offset` [extern]**20.68.5.6** `_submap_tile_offset` `uint8_t _submap_tile_offset` [extern]**20.68.5.7** `shadow_OAM` `volatile struct OAM_item_t shadow_OAM[]` [extern]

Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

**20.68.5.8** `_shadow_OAM_base` `uint8_t _shadow_OAM_base` [extern]

MSB of `shadow_OAM` address is used by OAM DMA copying routine

**20.69 gbdk-lib/include/nes/rgb\_to\_nes\_macro.h File Reference****Macros**

- `#define RGB_TO_NES(c)`

**20.69.1 Macro Definition Documentation****20.69.1.1** `RGB_TO_NES` `#define RGB_TO_NES(  
c )`**20.70 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` (void) `OLDCALL`
- `uint16_t` `randw` (void) `OLDCALL`
- void `initarand` (`uint16_t` seed) `OLDCALL`
- `uint8_t` `arand` (void) `OLDCALL`

## Variables

- `uint16_t` `__rand_seed`

### 20.70.1 Detailed Description

Random generator using the linear congruential method

Author

Luc Van den Borre

### 20.70.2 Macro Definition Documentation

**20.70.2.1 `RAND_MAX`** `#define RAND_MAX 255`

**20.70.2.2 `RANDW_MAX`** `#define RANDW_MAX 65535`

### 20.70.3 Function Documentation

**20.70.3.1 `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. One way to do this is sampling (`DIV_REG`) up to 2 times (high byte of seed value then the low byte) at variable, non-deterministic points in time (such as when the player presses buttons on the title screen or in a menu). It only needs to be called once to be initialized.

See also

`rand()`, `randw()`

**20.70.3.2 `rand()`** `uint8_t rand (`  
     `void )`

Returns a random byte (8 bit) value.

`initrand()` should be used to initialize the random number generator before using `rand()`

**20.70.3.3 `randw()`** `uint16_t randw (`  
     `void )`

Returns a random word (16 bit) value.

[initrand\(\)](#) should be used to initialize the random number generator before using [rand\(\)](#)

**20.70.3.4 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\(\)](#)

**20.70.3.5 arand()** `uint8_t arand (`  
`void )`

Returns a random number generated with the linear lagged additive method.

[initrand\(\)](#) should be used to initialize the random number generator before using [arand\(\)](#)

## 20.70.4 Variable Documentation

**20.70.4.1 \_\_rand\_seed** `uint16_t __rand_seed [extern]`

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;
```

## 20.71 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`

### 20.71.1 Macro Definition Documentation

**20.71.1.1 SP\_SIZE** `#define SP_SIZE 1`

**20.71.1.2 BP\_SIZE** `#define BP_SIZE 0`

**20.71.1.3 SPX\_SIZE** `#define SPX_SIZE 0`

**20.71.1.4 BPX\_SIZE** `#define BPX_SIZE SPX_SIZE`

**20.71.1.5 RET\_SIZE** `#define RET_SIZE 2`

**20.71.1.6 setjmp** `#define setjmp(  
 jump_buf ) __setjmp(jump_buf)`

## 20.71.2 Typedef Documentation

**20.71.2.1 jmp\_buf** `typedef unsigned char jmp_buf[RET_SIZE+SP_SIZE+BP_SIZE+SPX_SIZE+BPX_SIZE]`

## 20.71.3 Function Documentation

**20.71.3.1 \_\_setjmp()** `int __setjmp (  
 jmp_buf )`

**20.71.3.2 longjmp()** `_Noreturn void longjmp (  
 jmp_buf ,  
 int )`

## 20.72 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 [SYSTEM\\_60HZ](#) 0x00
- #define [SYSTEM\\_50HZ](#) 0x01
- #define [VBK\\_REG\\_VDP\\_ATTR\\_SHIFT](#)
- #define [J\\_UP](#) 0b00000001
- #define [J\\_DOWN](#) 0b00000010

- `#define J_LEFT 0b00000100`
- `#define J_RIGHT 0b00001000`
- `#define J_B 0b00010000`
- `#define J_A 0b00100000`
- `#define J_START 0b01000000`
- `#define J_SELECT 0b10000000`
- `#define M_TEXT_OUT 0x02U`
- `#define M_TEXT_INOUT 0x03U`
- `#define M_NO_SCROLL 0x04U`
- `#define M_NO_INTERP 0x08U`
- `#define S_BANK 0x01U`
- `#define S_FLIPX 0x02U`
- `#define S_FLIPY 0x04U`
- `#define S_PALETTE 0x08U`
- `#define S_PRIORITY 0x10U`
- `#define S_PAL(n) (((n) & 0x01U) << 3)`
- `#define __WRITE_VDP_REG_UNSAFE(REG, v) shadow_##REG=(v),VDP_CMD=(shadow_##REG),VDP←_CMD=REG`
- `#define __WRITE_VDP_REG(REG, v) shadow_##REG=(v);__asm__("di");VDP_CMD=(shadow_←##REG);VDP_CMD=REG;__asm__("ei")`
- `#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 SET_BORDER_COLOR(C) __WRITE_VDP_REG(VDP_R7, ((C) | 0xf0u))`
- `#define SHOW_BKG`
- `#define HIDE_BKG`
- `#define SHOW_WIN`
- `#define HIDE_WIN`
- `#define SHOW_SPRITES (_sprites_OFF = 0)`
- `#define HIDE_SPRITES (_sprites_OFF = 1)`
- `#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 DIV_REG get_r_reg()`
- `#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_CONTROL&(~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 HARDWARE_SPRITE_CAN_FLIP_X 0`
- `#define HARDWARE_SPRITE_CAN_FLIP_Y 0`
- `#define set_bkg_tile_xy set_tile_xy`
- `#define set_win_tile_xy set_tile_xy`
- `#define set_bkg_attribute_xy set_attribute_xy`
- `#define set_win_attribute_xy set_attribute_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 (void) OLDCALL`
- `uint8_t get_system (void)`
- `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)`
- `void move_bkg (uint8_t x, uint8_t y)`
- `void scroll_bkg (int8_t x, int8_t y)`
- `void vsync (void) PRESERVES_REGS(b)`
- `void wait_vbl_done (void) PRESERVES_REGS(b)`
- `void display_off (void)`
- `void refresh_OAM (void)`



- [uint8\\_t get\\_r\\_reg](#) (void) PRESERVES\_REGS(b)
- [void delay](#) (uint16\_t d) Z88DK\_FASTCALL
- [uint8\\_t joypad](#) (void) OLDCALL PRESERVES\_REGS(b)
- [uint8\\_t waitpad](#) (uint8\_t mask) Z88DK\_FASTCALL PRESERVES\_REGS(d)
- [void waitpadup](#) (void) PRESERVES\_REGS(d)
- [uint8\\_t joypad\\_init](#) (uint8\_t npads, joypads\_t \*joypads) Z88DK\_CALLEE
- [void joypad\\_ex](#) (joypads\_t \*joypads) Z88DK\_FASTCALL PRESERVES\_REGS(iyh)
- [void enable\\_interrupts](#) (void) PRESERVES\_REGS(a)
- [void disable\\_interrupts](#) (void) PRESERVES\_REGS(a)
- [void set\\_default\\_palette](#) (void)
- [void cgb\\_compatibility](#) (void)
- [void cpu\\_fast](#) (void)
- [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, const palette\_color\_t \*rgb\_data) Z88DK\_CALLEE
- [void set\\_native\\_tile\\_data](#) (uint16\_t start, uint16\_t ntiles, const void \*src) PRESERVES\_REGS(iyh)
- [void set\\_bkg\\_4bpp\\_data](#) (uint16\_t start, uint16\_t ntiles, const void \*src) PRESERVES\_REGS(iyh)
- [void set\\_bkg\\_native\\_data](#) (uint16\_t start, uint16\_t ntiles, const void \*src) PRESERVES\_REGS(iyh)
- [void set\\_sprite\\_4bpp\\_data](#) (uint8\_t start, uint16\_t ntiles, const void \*src) PRESERVES\_REGS(iyh)
- [void set\\_sprite\\_native\\_data](#) (uint8\_t start, uint16\_t ntiles, const void \*src) PRESERVES\_REGS(iyh)
- [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
- [void set\\_tile\\_map\\_compat](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*tiles) Z88DK\_CALLEE
- [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\\_bkg\\_attributes](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*tiles)
- [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
- [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
- [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 set\\_bkg\\_submap\\_attributes](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint8\_t \*map, uint8\_t map\_w)
- [void fill\\_rect](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint16\_t tile) Z88DK\_CALLEE
- [void fill\\_rect\\_compat](#) (uint8\_t x, uint8\_t y, uint8\_t w, uint8\_t h, const uint16\_t tile) Z88DK\_CALLEE
- [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 * set_attribute_xy (uint8_t x, uint8_t y, uint8_t a) Z88DK_CALLEE PRESERVES_REGS(iyh`
- `uint8_t * get_bkg_xy_addr (uint8_t x, uint8_t y) Z88DK_CALLEE PRESERVES_REGS(iyh`

## Variables

- `const UBYTE _BIOS`
- `const uint8_t _SYSTEM`
- `void c`
- `void d`
- `void e`
- `void iyh`
- `void iyl`
- `void h`
- `void l`
- `volatile uint16_t sys_time`
- `void b`
- `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`
- `volatile uint8_t _sprites_OFF`

### 20.72.1 Detailed Description

SMS/GG specific functions.

### 20.72.2 Macro Definition Documentation

**20.72.2.1 SEGA** `#define SEGA`

**20.72.2.2 SYSTEM\_60HZ** `#define SYSTEM_60HZ 0x00`

**20.72.2.3 SYSTEM\_50HZ** `#define SYSTEM_50HZ 0x01`

**20.72.2.4 VBK\_REG** `#define VBK_REG VDP_ATTR_SHIFT`

**20.72.2.5 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](#)

**20.72.2.6 J\_DOWN** `#define J_DOWN 0b00000010`**20.72.2.7 J\_LEFT** `#define J_LEFT 0b00000100`**20.72.2.8 J\_RIGHT** `#define J_RIGHT 0b00001000`**20.72.2.9 J\_B** `#define J_B 0b00010000`**20.72.2.10 J\_A** `#define J_A 0b00100000`**20.72.2.11 J\_START** `#define J_START 0b01000000`**20.72.2.12 J\_SELECT** `#define J_SELECT 0b10000000`**20.72.2.13 M\_TEXT\_OUT** `#define M_TEXT_OUT 0x02U`

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

**20.72.2.14 M\_TEXT\_INOUT** `#define M_TEXT_INOUT 0x03U`**20.72.2.15 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\(\)](#)

**20.72.2.16 M\_NO\_INTERP** `#define M_NO_INTERP 0x08U`

Set this to disable interpretation

See also

[mode\(\)](#)

**20.72.2.17 S\_BANK** `#define S_BANK 0x01U`  
The ninth bit of the tile id

**20.72.2.18 S\_FLIPX** `#define S_FLIPX 0x02U`  
If set the background tile will be flipped horizontally.

**20.72.2.19 S\_FLIPY** `#define S_FLIPY 0x04U`  
If set the background tile will be flipped vertically.

**20.72.2.20 S\_PALETTE** `#define S_PALETTE 0x08U`  
If set the background tile palette.

**20.72.2.21 S\_PRIORITY** `#define S_PRIORITY 0x10U`  
If set the background tile priority.

**20.72.2.22 S\_PAL** `#define S_PAL(`  
`n ) ((n) & 0x01U) << 3)`  
Dummy function used by other platforms. Required for the png2asset tool's metasprite output.

**20.72.2.23 \_\_WRITE\_VDP\_REG\_UNSAFE** `#define __WRITE_VDP_REG_UNSAFE(`  
`REG,`  
`v ) shadow_##REG=(v), VDP_CMD=(shadow_##REG), VDP_CMD=REG`

**20.72.2.24 \_\_WRITE\_VDP\_REG** `#define __WRITE_VDP_REG(`  
`REG,`  
`v ) shadow_##REG=(v); __asm__ ("di"); VDP_CMD=(shadow_##REG); VDP_CMD=REG; __asm__ ↵`  
`("ei")`

**20.72.2.25 \_\_READ\_VDP\_REG** `#define __READ_VDP_REG(`  
`REG ) shadow_##REG`

**20.72.2.26 EMPTY\_IFLAG** `#define EMPTY_IFLAG 0x00U`  
Disable calling of interrupt service routines

**20.72.2.27 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](#)

**20.72.2.28 LCD\_IFLAG** `#define LCD_IFLAG 0x02U`  
LCD Interrupt when triggered by the STAT register.

See also

[set\\_interrupts\(\)](#),  
[add\\_LCD](#)

**20.72.2.29 TIM\_IFLAG** `#define TIM_IFLAG 0x04U`  
Does nothing on SMS/GG

**20.72.2.30 SIO\_IFLAG** `#define SIO_IFLAG 0x08U`  
Does nothing on SMS/GG

**20.72.2.31 JOY\_IFLAG** `#define JOY_IFLAG 0x10U`  
Does nothing on SMS/GG

**20.72.2.32 SCREENWIDTH** `#define SCREENWIDTH DEVICE_SCREEN_PX_WIDTH`  
Width of the visible screen in pixels.

**20.72.2.33 SCREENHEIGHT** `#define SCREENHEIGHT DEVICE_SCREEN_PX_HEIGHT`  
Height of the visible screen in pixels.

**20.72.2.34 MINWNDPOSX** `#define MINWNDPOSX 0x00U`  
The Minimum X position of the Window Layer (Left edge of screen)

See also

[move\\_win\(\)](#)

**20.72.2.35 MINWNDPOSY** `#define MINWNDPOSY 0x00U`  
The Minimum Y position of the Window Layer (Top edge of screen)

See also

[move\\_win\(\)](#)

**20.72.2.36 MAXWNDPOSX** `#define MAXWNDPOSX 0x00U`  
The Maximum X position of the Window Layer (Right edge of screen)

See also

[move\\_win\(\)](#)

**20.72.2.37 MAXWNDPOSY** `#define MAXWNDPOSY 0x00U`  
The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

[move\\_win\(\)](#)

**20.72.2.38 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](#)

**20.72.2.39 DISPLAY\_OFF** `#define DISPLAY_OFF display_off();`  
Turns the display off immediately.

See also

[display\\_off](#), [DISPLAY\\_ON](#)

**20.72.2.40 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](#)

**20.72.2.41 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](#)

**20.72.2.42 SET\_BORDER\_COLOR** `#define SET_BORDER_COLOR(C) __WRITE_VDP_REG(VDP_R7, ((C) | 0xf0u))`  
Sets border color

**20.72.2.43 SHOW\_BKG** `#define SHOW_BKG`  
Turns on the background layer. Not yet implemented

**20.72.2.44 HIDE\_BKG** `#define HIDE_BKG`  
Turns off the background layer. Not yet implemented

**20.72.2.45 SHOW\_WIN** `#define SHOW_WIN`  
Turns on the window layer Not yet implemented

**20.72.2.46 HIDE\_WIN** `#define HIDE_WIN`  
Turns off the window layer. Not yet implemented

**20.72.2.47 SHOW\_SPRITES** `#define SHOW_SPRITES (_sprites_OFF = 0)`  
Turns on the sprites layer.

**20.72.2.48 HIDE\_SPRITES** `#define HIDE_SPRITES (_sprites_OFF = 1)`  
Turns off the sprites layer.

**20.72.2.49 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.

**20.72.2.50 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.

**20.72.2.51 DEVICE\_SUPPORTS\_COLOR** `#define DEVICE_SUPPORTS_COLOR (TRUE)`

Macro returns TRUE if device supports color (it always does on SMS/GG)

**20.72.2.52 DIV\_REG** `#define DIV_REG get_r_reg()`**20.72.2.53 \_current\_bank** `#define _current_bank MAP_FRAME1`

Tracks current active ROM bank in frame 1

**20.72.2.54 CURRENT\_BANK** `#define CURRENT_BANK MAP_FRAME1`**20.72.2.55 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\(\)](#)

**20.72.2.56 BANKREF** `#define BANKREF( VARNAME )`**Value:**

```
void __func_ ## VARNAME(void) __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.

**20.72.2.57 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 <a href="#">BANKREF()</a> |
|----------------|----------------------------------------------------------|

This makes a [BANKREF\(\)](#) reference in another source file accessible in the current file for use with [BANK\(\)](#).

See also

[BANKREF\(\)](#), [BANK\(\)](#)

**20.72.2.58 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 |
|----------|-----------------------|

**20.72.2.59 SWITCH\_ROM1** `#define SWITCH_ROM1 SWITCH\_ROM`

**20.72.2.60 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 |
|----------|-----------------------|

**20.72.2.61 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 |
|----------|------------------------|

**20.72.2.62 ENABLE\_RAM** `#define ENABLE_RAM RAM_CONTROL|=RAMCTL\_RAM`

Enables RAM

**20.72.2.63 DISABLE\_RAM** `#define DISABLE_RAM RAM_CONTROL&= (~RAMCTL\_RAM)`

Disables RAM

**20.72.2.64 set\_bkg\_palette\_entry** `#define set_bkg_palette_entry set\_palette\_entry`

**20.72.2.65 set\_sprite\_palette\_entry** `#define set_sprite_palette_entry(  
    palette,  
    entry,  
    rgb_data ) set\_palette\_entry(1,entry,rgb_data)`



**20.72.2.66 set\_bkg\_palette** #define set\_bkg\_palette [set\\_palette](#)

**20.72.2.67 set\_sprite\_palette** #define set\_sprite\_palette(  
    *first\_palette*,  
    *nb\_palettes*,  
    *rgb\_data* ) [set\\_palette](#)(1,1,rgb\_data)

**20.72.2.68 COMPAT\_PALETTE** #define COMPAT\_PALETTE(  
    *C0*,  
    *C1*,  
    *C2*,  
    *C3* ) (([uint16\\_t](#)) (*C3*) << 12) | (([uint16\\_t](#)) (*C2*) << 8) | (([uint16\\_t](#)) (*C1*) << 4) |  
([uint16\\_t](#)) (*C0*) )

**20.72.2.69 set\_bkg\_tiles** #define set\_bkg\_tiles [set\\_tile\\_map\\_compat](#)

**20.72.2.70 set\_win\_tiles** #define set\_win\_tiles [set\\_tile\\_map\\_compat](#)

**20.72.2.71 fill\_bkg\_rect** #define fill\_bkg\_rect [fill\\_rect\\_compat](#)

**20.72.2.72 fill\_win\_rect** #define fill\_win\_rect [fill\\_rect\\_compat](#)

**20.72.2.73 DISABLE\_VBL\_TRANSFER** #define DISABLE\_VBL\_TRANSFER [\\_shadow\\_OAM\\_base](#) = 0  
Disable shadow OAM to VRAM copy on each VBlank

**20.72.2.74 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

**20.72.2.75 MAX\_HARDWARE\_SPRITES** #define MAX\_HARDWARE\_SPRITES 64  
Amount of hardware sprites in OAM

**20.72.2.76 HARDWARE\_SPRITE\_CAN\_FLIP\_X** #define HARDWARE\_SPRITE\_CAN\_FLIP\_X 0  
True if sprite hardware can flip sprites by X (horizontally)

**20.72.2.77 HARDWARE\_SPRITE\_CAN\_FLIP\_Y** #define HARDWARE\_SPRITE\_CAN\_FLIP\_Y 0  
True if sprite hardware can flip sprites by Y (vertically)

**20.72.2.78 set\_bkg\_tile\_xy** #define set\_bkg\_tile\_xy [set\\_tile\\_xy](#)

**20.72.2.79 set\_win\_tile\_xy** #define set\_win\_tile\_xy [set\\_tile\\_xy](#)

**20.72.2.80 set\_bkg\_attribute\_xy** #define set\_bkg\_attribute\_xy [set\\_attribute\\_xy](#)

**20.72.2.81 set\_win\_attribute\_xy** `#define set_win_attribute_xy set_attribute_xy`

**20.72.2.82 get\_win\_xy\_addr** `#define get_win_xy_addr get_bkg_xy_addr`

### 20.72.3 Typedef Documentation

**20.72.3.1 int\_handler** `typedef void(* int_handler) (void) NONBANKED`  
Interrupt handlers

### 20.72.4 Function Documentation

**20.72.4.1 WRITE\_VDP\_CMD()** `void WRITE_VDP_CMD (`  
    `uint16_t cmd )`

**20.72.4.2 WRITE\_VDP\_DATA()** `void WRITE_VDP_DATA (`  
    `uint16_t data )`

**20.72.4.3 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](#)

**20.72.4.4 get\_mode()** `uint8_t get_mode (`  
    `void )`

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](#)

**20.72.4.5 get\_system()** `uint8_t get_system (`  
    `void ) [inline]`

Returns the system gbdk is running on.

**20.72.4.6 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](#)

**20.72.4.7 remove\_VBL()** `void remove_VBL (`  
`int_handler h )`

Removes the VBL interrupt handler.

## See also

[add\\_VBL\(\)](#)

**20.72.4.8 remove\_LCD()** `void remove_LCD (`  
`int_handler h )`

Removes the LCD interrupt handler.

## See also

[add\\_LCD\(\)](#), [remove\\_VBL\(\)](#)

**20.72.4.9 remove\_TIM()** `void remove_TIM (`  
`int_handler h )`

**20.72.4.10 remove\_SIO()** `void remove_SIO (`  
`int_handler h )`

**20.72.4.11 remove\_JOY()** `void remove_JOY (`  
`int_handler h )`

**20.72.4.12 add\_VBL()** `void add_VBL (`  
`int_handler h )`

Adds a V-blank interrupt handler.

**20.72.4.13 add\_LCD()** `void add_LCD (`  
`int_handler h )`

Adds a LCD interrupt handler.

**20.72.4.14 add\_TIM()** `void add_TIM (`  
`int_handler h )`

Does nothing on SMS/GG

**20.72.4.15 add\_SIO()** `void add_SIO (`  
    `int_handler h )`

Does nothing on SMS/GG

**20.72.4.16 add\_JOY()** `void add_JOY (`  
    `int_handler h )`

Does nothing on SMS/GG

**20.72.4.17 cancel\_pending\_interrupts()** `uint8_t cancel_pending_interrupts (`  
    `void ) [inline]`

Cancel pending interrupts

**20.72.4.18 move\_bkg()** `void move_bkg (`  
    `uint8_t x,`  
    `uint8_t y ) [inline]`

**20.72.4.19 scroll\_bkg()** `void scroll_bkg (`  
    `int8_t x,`  
    `int8_t y ) [inline]`

**20.72.4.20 vsync()** `void vsync (`  
    `void )`

HALTs the CPU and waits for the vertical blank interrupt.

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.

**20.72.4.21 wait\_vbl\_done()** `void wait_vbl_done (`  
    `void )`

Obsolete. This function has been replaced by [vsync\(\)](#), which has identical behavior.

**20.72.4.22 display\_off()** `void display_off (`  
    `void ) [inline]`

Turns the display off.

See also

[DISPLAY\\_ON](#)

**20.72.4.23 refresh\_OAM()** `void refresh_OAM (`  
    `void )`

Copies data from shadow OAM to OAM

**20.72.4.24 get\_r\_reg()** `uint8_t get_r_reg (`  
    `void )`

Return R register for the DIV\_REG emulation

Increments once per CPU instruction (fetches the Z80 CPU R register)

**20.72.4.25 delay()** `void delay (`  
    `uint16_t d )`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled

**20.72.4.26** `joypad()` `uint8_t` `joypad` (  
`void` )

Reads and returns the current state of the joypad.

**20.72.4.27** `waitpad()` `uint8_t` `waitpad` (  
`uint8_t` *mask* )

Waits until at least one of the buttons given in mask are pressed.

**20.72.4.28** `waitpadup()` `void` `waitpadup` (  
`void` )

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.

**20.72.4.29** `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 <code>joypads_t</code> 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`

**20.72.4.30** `joypad_ex()` `void` `joypad_ex` (  
`joypads_t` \* *joypads* )

Polls all available joypads

#### Parameters

|                |                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>joypads</i> | 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`

**20.72.4.31** `enable_interrupts()` `void` `enable_interrupts` (  
`void` ) `[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](#)

**20.72.4.32 `disable_interrupts()`** `void disable_interrupts (`  
`void ) [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](#)

**20.72.4.33 `set_default_palette()`** `void set_default_palette (`  
`void )`

**20.72.4.34 `cgb_compatibility()`** `void cgb_compatibility (`  
`void ) [inline]`

Obsolete. This function has been replaced by [set\\_default\\_palette\(\)](#), which has identical behavior.

**20.72.4.35 `cpu_fast()`** `void cpu_fast (`  
`void ) [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](#)

**20.72.4.36 `set_palette_entry()`** `void set_palette_entry (`  
`uint8_t palette,`  
`uint8_t entry,`  
`uint16_t rgb_data )`

**20.72.4.37 `set_palette()`** `void set_palette (`  
`uint8_t first_palette,`  
`uint8_t nb_palettes,`  
`const palette_color_t * rgb_data )`

Set color palette(s)

## Parameters

|                      |                                                                 |
|----------------------|-----------------------------------------------------------------|
| <i>first_palette</i> | Index of the first 16 color palette to write (0-1)              |
| <i>nb_palettes</i>   | Number of palettes to write (1-2, max depends on first_palette) |
| <i>rgb_data</i>      | Pointer to source palette data                                  |

Writes **nb\_palettes** to palette data starting at **first\_palette**, Palette data is sourced from **rgb\_data**.

- Palette 0 can be used for the Background.
- Palette 1 is shared between Background and Sprites.

## On the Game Gear

- Each Palette is 32 bytes in size: 16 colors x 2 bytes per palette color entry.
- Each color (16 per palette) is packed as BGR-444 format (x:4:4:4, MSBits [15..12] are unused).
- Each component (R, G, B) may have values from 0 - 15 (4 bits), 15 is brightest.

## On the SMS

- On SMS each Palette is 16 bytes in size: 16 colors x 1 byte per palette color entry.
- Each color (16 per palette) is packed as BGR-222 format (x:2:2:2, MSBits [7..6] are unused).
- Each component (R, G, B) may have values from 0 - 3 (2 bits), 3 is brightest.

## See also

[RGB\(\)](#), [set\\_sprite\\_palette\(\)](#), [set\\_bkg\\_palette\(\)](#), [set\\_palette\\_entry\(\)](#), [set\\_sprite\\_palette\\_entry\(\)](#), [set\\_bkg\\_palette\\_entry\(\)](#), [set\\_sprite\\_palette\(\)](#)

**20.72.4.38 set\_native\_tile\_data()** void set\_native\_tile\_data (   
     uint16\_t start,   
     uint16\_t ntiles,   
     const void \* src )

**20.72.4.39 set\_bkg\_4bpp\_data()** void set\_bkg\_4bpp\_data (   
     uint16\_t start,   
     uint16\_t ntiles,   
     const void \* src )

**20.72.4.40 set\_bkg\_native\_data()** void set\_bkg\_native\_data (   
     uint16\_t start,   
     uint16\_t ntiles,   
     const void \* src )

**20.72.4.41 set\_sprite\_4bpp\_data()** void set\_sprite\_4bpp\_data (   
     uint8\_t start,   
     uint16\_t ntiles,   
     const void \* src )

**20.72.4.42** **set\_sprite\_native\_data()** void set\_sprite\_native\_data (  
    uint8\_t start,  
    uint16\_t ntiles,  
    const void \* src )

**20.72.4.43** **set\_2bpp\_palette()** void set\_2bpp\_palette (  
    uint16\_t palette ) [inline]

**20.72.4.44** **set\_tile\_2bpp\_data()** void set\_tile\_2bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src,  
    uint16\_t palette )

**20.72.4.45** **set\_bkg\_data()** void set\_bkg\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.46** **set\_sprite\_data()** void set\_sprite\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.47** **set\_bkg\_2bpp\_data()** void set\_bkg\_2bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.48** **set\_sprite\_2bpp\_data()** void set\_sprite\_2bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.49** **set\_1bpp\_colors()** void set\_1bpp\_colors (  
    uint8\_t fgcolor,  
    uint8\_t bgcolor ) [inline]

**20.72.4.50** **set\_tile\_1bpp\_data()** void set\_tile\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src,  
    uint16\_t colors )



**20.72.4.51 set\_bkg\_1bpp\_data()** void set\_bkg\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.52 set\_sprite\_1bpp\_data()** void set\_sprite\_1bpp\_data (  
    uint16\_t start,  
    uint16\_t ntiles,  
    const void \* src ) [inline]

**20.72.4.53 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**.

**20.72.4.54 vmemcpy()** void vmemcpy (  
    uint16\_t dst,  
    const void \* src,  
    uint16\_t size )

**20.72.4.55 set\_tile\_map()** void set\_tile\_map (  
    uint8\_t x,  
    uint8\_t y,  
    uint8\_t w,  
    uint8\_t h,  
    const uint8\_t \* tiles )

**20.72.4.56 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 )

**20.72.4.57 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]

**20.72.4.58 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]

```

**20.72.4.59 set\_bkg\_attributes()** void set\_bkg\_attributes (

```

uint8_t x,
uint8_t y,
uint8_t w,
uint8_t h,
const uint8_t * tiles ) [inline]

```

**20.72.4.60 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 )

```

**20.72.4.61 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 )

```

**20.72.4.62 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> <sub>↔</sub><br><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](#)

#### 20.72.4.63 [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](#)

**20.72.4.64 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]`

**20.72.4.65 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]`

**20.72.4.66 set\_bkg\_submap\_attributes()** `void set_bkg_submap_attributes (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t w,`  
    `uint8_t h,`  
    `const uint8_t * map,`  
    `uint8_t map_w ) [inline]`

**20.72.4.67 fill\_rect()** `void fill_rect (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t w,`  
    `uint8_t h,`  
    `const uint16_t tile )`

**20.72.4.68 fill\_rect\_compat()** `void fill_rect_compat (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t w,`  
    `uint8_t h,`  
    `const uint16_t tile )`

**20.72.4.69 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

**20.72.4.70 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<br>In CGB Mode this could be either in VRAM Bank<br>0 or 1, depending on Bit 3 of the OAM Attribute Flag<br>(see <a href="#">set_sprite_prop</a> ) |

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](#)

**20.72.4.71 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

**20.72.4.72 set\_sprite\_prop()** `void set_sprite_prop (`  
    `uint8_t nb,`  
    `uint8_t prop ) [inline]`

Function has no affect on sms.

This function is only here to enable game portability

**20.72.4.73 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

**20.72.4.74 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).<br>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).<br>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.

**20.72.4.75 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 <b>X axis</b><br>Range: -128 - 127 |
| <i>y</i>  | Number of pixels to move the sprite on the <b>Y axis</b><br>Range: -128 - 127 |

#### See also

[move\\_sprite](#) for more details about the X and Y position

**20.72.4.76 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 |
|-----------|-----------------------------|

**20.72.4.77 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               |

**20.72.4.78 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

**20.72.4.79 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

**20.72.4.80 set\_attribute\_xy()** `uint8_t* set_attribute_xy (`  
    `uint8_t x,`  
    `uint8_t y,`  
    `uint8_t a ) [inline]`

Set single attribute data *a* on background layer at *x*,*y*

#### Parameters

|          |                 |
|----------|-----------------|
| <i>x</i> | X-coordinate    |
| <i>y</i> | Y-coordinate    |
| <i>a</i> | tile attributes |

#### Returns

returns the address of tile attribute, so you may use faster [set\\_vram\\_byte\(\)](#) later

**20.72.4.81 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

## 20.72.5 Variable Documentation

**20.72.5.1** `_BIOS` `const UBYTE _BIOS` [extern]

**20.72.5.2** `_SYSTEM` `const uint8_t _SYSTEM` [extern]

**20.72.5.3** `c` `void c`

**20.72.5.4** `d` `void d`

**20.72.5.5** `e` `void e`

**20.72.5.6** `iyh` `void iyh`

**20.72.5.7** `iyi` `uint8_t iyi`

Initial value:

```
{  
    __asm__("ei")
```

**20.72.5.8** `h` `void h`

**20.72.5.9** `l` `void l`

**20.72.5.10** `sys_time` `volatile uint16_t sys_time` [extern]

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)

**20.72.5.11** `b` `void b`

**20.72.5.12** `_current_2bpp_palette` `uint16_t _current_2bpp_palette` [extern]

**20.72.5.13** `_current_1bpp_colors` `uint16_t _current_1bpp_colors` [extern]

**20.72.5.14** `_map_tile_offset` `uint8_t _map_tile_offset` [extern]



**20.72.5.15** `_submap_tile_offset` `uint8_t _submap_tile_offset` [extern]

**20.72.5.16** `shadow_OAM` `volatile uint8_t shadow_OAM[]` [extern]  
Shadow OAM array in WRAM, that is transferred into the real OAM each VBlank

**20.72.5.17** `_shadow_OAM_base` `volatile uint8_t _shadow_OAM_base` [extern]  
MSB of shadow\_OAM address is used by OAM copying routine  
MSB of shadow\_OAM address is used by OAM DMA copying routine

**20.72.5.18** `_shadow_OAM_OFF` `volatile uint8_t _shadow_OAM_OFF` [extern]  
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](#)

**20.72.5.19** `_sprites_OFF` `volatile uint8_t _sprites_OFF` [extern]

## 20.73 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)

### 20.73.1 Function Documentation

**20.73.1.1** `atomic_flag_test_and_set()` `_Bool atomic_flag_test_and_set (`  
`volatile atomic_flag * object )`

**20.73.1.2 atomic\_flag\_clear()** `void atomic_flag_clear (`  
`volatile atomic_flag * object )`

## 20.74 gbk-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`

### 20.74.1 Macro Definition Documentation

**20.74.1.1 true** `#define true ((_Bool)+1)`

**20.74.1.2 false** `#define false ((_Bool)+0)`

**20.74.1.3 bool** `#define bool _Bool`

**20.74.1.4 \_\_bool\_true\_false\_are\_defined** `#define __bool_true_false_are_defined 1`

## 20.75 gbk-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`

### 20.75.1 Macro Definition Documentation

**20.75.1.1 NULL** `#define NULL (void *)0`

**20.75.1.2 \_\_PTRDIFF\_T\_DEFINED** `#define __PTRDIFF_T_DEFINED`

**20.75.1.3 \_\_SIZE\_T\_DEFINED** `#define __SIZE_T_DEFINED`

**20.75.1.4** `__WCHAR_T_DEFINED` `#define __WCHAR_T_DEFINED`

**20.75.1.5** `offsetof` `#define offsetof(  
    s,  
    m ) __builtin_offsetof (s, m)`

## 20.75.2 Typedef Documentation

**20.75.2.1** `ptrdiff_t` `typedef int ptrdiff_t`

**20.75.2.2** `size_t` `typedef unsigned int size_t`

**20.75.2.3** `wchar_t` `typedef unsigned long int wchar_t`

## 20.76 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`

## 20.76.1 Macro Definition Documentation

**20.76.1.1 INT8\_MIN** `#define INT8_MIN (-128)`

**20.76.1.2 INT16\_MIN** `#define INT16_MIN (-32767-1)`

**20.76.1.3 INT32\_MIN** `#define INT32_MIN (-2147483647L-1)`

**20.76.1.4 INT8\_MAX** `#define INT8_MAX (127)`

**20.76.1.5 INT16\_MAX** `#define INT16_MAX (32767)`

**20.76.1.6 INT32\_MAX** `#define INT32_MAX (2147483647L)`

**20.76.1.7 UINT8\_MAX** `#define UINT8_MAX (255)`

**20.76.1.8 UINT16\_MAX** `#define UINT16_MAX (65535)`

**20.76.1.9 UINT32\_MAX** `#define UINT32_MAX (4294967295UL)`

**20.76.1.10 INT\_LEAST8\_MIN** `#define INT_LEAST8_MIN INT8_MIN`

**20.76.1.11 INT\_LEAST16\_MIN** `#define INT_LEAST16_MIN INT16_MIN`

**20.76.1.12 INT\_LEAST32\_MIN** `#define INT_LEAST32_MIN INT32_MIN`

**20.76.1.13 INT\_LEAST8\_MAX** `#define INT_LEAST8_MAX INT8_MAX`

**20.76.1.14 INT\_LEAST16\_MAX** `#define INT_LEAST16_MAX INT16_MAX`

**20.76.1.15 INT\_LEAST32\_MAX** `#define INT_LEAST32_MAX INT32_MAX`

**20.76.1.16 UINT\_LEAST8\_MAX** `#define UINT_LEAST8_MAX UINT8_MAX`

**20.76.1.17 UINT\_LEAST16\_MAX** `#define UINT_LEAST16_MAX UINT16_MAX`

**20.76.1.18 UINT\_LEAST32\_MAX** `#define UINT_LEAST32_MAX UINT32_MAX`

**20.76.1.19 INT\_FAST8\_MIN** `#define INT_FAST8_MIN INT8_MIN`

**20.76.1.20 INT\_FAST16\_MIN** `#define INT_FAST16_MIN INT16_MIN`

**20.76.1.21 INT\_FAST32\_MIN** `#define INT_FAST32_MIN INT32_MIN`

**20.76.1.22 INT\_FAST8\_MAX** `#define INT_FAST8_MAX INT8_MAX`

**20.76.1.23 INT\_FAST16\_MAX** `#define INT_FAST16_MAX INT16_MAX`

**20.76.1.24 INT\_FAST32\_MAX** `#define INT_FAST32_MAX INT32_MAX`

**20.76.1.25 UINT\_FAST8\_MAX** `#define UINT_FAST8_MAX UINT8_MAX`

**20.76.1.26 UINT\_FAST16\_MAX** `#define UINT_FAST16_MAX UINT16_MAX`

**20.76.1.27 UINT\_FAST32\_MAX** `#define UINT_FAST32_MAX UINT32_MAX`

**20.76.1.28 INTPTR\_MIN** `#define INTPTR_MIN (-32767-1)`

**20.76.1.29 INTPTR\_MAX** `#define INTPTR_MAX (32767)`

**20.76.1.30 UINTPTR\_MAX** `#define UINTPTR_MAX (65535)`

**20.76.1.31 INTMAX\_MIN** `#define INTMAX_MIN (-2147483647L-1)`

**20.76.1.32 INTMAX\_MAX** `#define INTMAX_MAX (2147483647L)`

**20.76.1.33 UINTMAX\_MAX** `#define UINTMAX_MAX (4294967295UL)`

**20.76.1.34 PTRDIFF\_MIN** `#define PTRDIFF_MIN (-32767-1)`

**20.76.1.35 PTRDIFF\_MAX** `#define PTRDIFF_MAX (32767)`

**20.76.1.36 SIG\_ATOMIC\_MIN** `#define SIG_ATOMIC_MIN (0)`

**20.76.1.37 SIG\_ATOMIC\_MAX** `#define SIG_ATOMIC_MAX (255)`

**20.76.1.38 SIZE\_MAX** `#define SIZE_MAX (65535u)`

**20.76.1.39 INT8\_C** `#define INT8_C(  
    c ) c`

**20.76.1.40 INT16\_C** `#define INT16_C(  
    c ) c`

**20.76.1.41 INT32\_C** `#define INT32_C(  
    c ) c ## L`

**20.76.1.42 UINT8\_C** `#define UINT8_C(  
    c ) c ## U`

**20.76.1.43 UINT16\_C** `#define UINT16_C(  
    c ) c ## U`

**20.76.1.44 UINT32\_C** `#define UINT32_C(  
    c ) c ## UL`

**20.76.1.45 WCHAR\_MIN** `#define WCHAR_MIN 0`

**20.76.1.46 WCHAR\_MAX** `#define WCHAR_MAX 0xffffffff`

**20.76.1.47 WINT\_MIN** `#define WINT_MIN 0`

**20.76.1.48 WINT\_MAX** `#define WINT_MAX 0xffffffff`

**20.76.1.49 INTMAX\_C** `#define INTMAX_C(  
    c ) c ## L`

**20.76.1.50 UINTMAX\_C** `#define UINTMAX_C(  
    c ) c ## UL`

## 20.76.2 Typedef Documentation

**20.76.2.1 int8\_t** `typedef signed char int8_t`

**20.76.2.2 int16\_t** `typedef short int int16_t`

**20.76.2.3 int32\_t** `typedef long int int32_t`

**20.76.2.4 uint8\_t** `typedef unsigned char uint8_t`

**20.76.2.5** `uint16_t` typedef unsigned short int `uint16_t`

**20.76.2.6** `uint32_t` typedef unsigned long int `uint32_t`

**20.76.2.7** `int_least8_t` typedef signed char `int_least8_t`

**20.76.2.8** `int_least16_t` typedef short int `int_least16_t`

**20.76.2.9** `int_least32_t` typedef long int `int_least32_t`

**20.76.2.10** `uint_least8_t` typedef unsigned char `uint_least8_t`

**20.76.2.11** `uint_least16_t` typedef unsigned short int `uint_least16_t`

**20.76.2.12** `uint_least32_t` typedef unsigned long int `uint_least32_t`

**20.76.2.13** `int_fast8_t` typedef signed char `int_fast8_t`

**20.76.2.14** `int_fast16_t` typedef int `int_fast16_t`

**20.76.2.15** `int_fast32_t` typedef long int `int_fast32_t`

**20.76.2.16** `uint_fast8_t` typedef unsigned char `uint_fast8_t`

**20.76.2.17** `uint_fast16_t` typedef unsigned int `uint_fast16_t`

**20.76.2.18** `uint_fast32_t` typedef unsigned long int `uint_fast32_t`

**20.76.2.19** `intptr_t` typedef int `intptr_t`

**20.76.2.20** `uintptr_t` typedef unsigned int `uintptr_t`

**20.76.2.21** `intmax_t` typedef long int `intmax_t`

**20.76.2.22** `uintmax_t` typedef unsigned long int `uintmax_t`



## 20.77 gbdk-lib/include/stdio.h File Reference

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

### Functions

- void [putchar](#) (char *c*) [OLDCALL](#) REENTRANT
- void [printf](#) (const char \**format*,...)
- void [sprintf](#) (char \**str*, const char \**format*,...)
- void [puts](#) (const char \**s*)
- char \* [gets](#) (char \**s*) [OLDCALL](#)
- char [getchar](#) (void) [OLDCALL](#)

### 20.77.1 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.

### 20.77.2 Function Documentation

**20.77.2.1 putchar()** void putchar (  
char *c* )

Print char to stdout.

#### Parameters

|          |                    |
|----------|--------------------|
| <i>c</i> | Character to print |
|----------|--------------------|

**20.77.2.2 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.

**20.77.2.3 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 <a href="#">printf</a> |

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.

**20.77.2.4 puts()** `void puts (`  
`const char * s )`

[puts\(\)](#) writes the string **s** and a trailing newline to stdout.

**20.77.2.5 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**

**20.77.2.6 getchar()** `char getchar (`  
`void )`

[getchar\(\)](#) Reads and returns a single character from stdin.

## 20.78 gbk-lib/include/stdlib.h File Reference

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

### Functions

- void [exit](#) (int status) [OLDSCALL](#)
- int [abs](#) (int i)
- long [labs](#) (long num) [OLDSCALL](#)
- int [atoi](#) (const char \*s)
- long [atol](#) (const char \*s)
- char \* [itoa](#) (int n, char \*s, unsigned char radix) [OLDSCALL](#)
- char \* [uitoa](#) (unsigned int n, char \*s, unsigned char radix) [OLDSCALL](#)
- char \* [ltoa](#) (long n, char \*s, unsigned char radix) [OLDSCALL](#)
- char \* [ultoa](#) (unsigned long n, char \*s, unsigned char radix) [OLDSCALL](#)
- 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)

## 20.78.1 Function Documentation

**20.78.1.1 exit()** void exit (  
int status )

file stdlib.h 'Standard library' functions, for whatever that means. Causes normal program termination and the value of status is returned to the parent. All open streams are flushed and closed.

**20.78.1.2 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*.

**20.78.1.3 labs()** long labs (  
long num )

Returns the absolute value of long int **num**

### Parameters

|            |                                          |
|------------|------------------------------------------|
| <i>num</i> | Long integer to obtain absolute value of |
|------------|------------------------------------------|

**20.78.1.4 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

**20.78.1.5 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

**20.78.1.6 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

**20.78.1.7 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

**20.78.1.8 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

**20.78.1.9 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

**20.78.1.10 calloc()** void\* calloc (  
     size\_t nmemb,  
     size\_t size )

Memory allocation functions

**20.78.1.11 malloc()** void\* malloc (  
     size\_t size )

**20.78.1.12 realloc()** void\* realloc (  
     void \* ptr,  
     size\_t size )

**20.78.1.13 free()** void free (  
     void \* ptr )

**20.78.1.14 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   |
| <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.

**20.78.1.15 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 |

## 20.79 gbdk-lib/include/stdnoreturn.h File Reference

### Macros

- `#define noreturn _Noreturn`

#### 20.79.1 Macro Definition Documentation

**20.79.1.1 noreturn** `#define noreturn _Noreturn`

## 20.80 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` (void) `OLDCALL`
- `time_t time` (`time_t *`t)

#### 20.80.1 Detailed Description

Sort of ANSI compliant time functions.

#### 20.80.2 Macro Definition Documentation

**20.80.2.1 CLOCKS\_PER\_SEC** `#define CLOCKS_PER_SEC 60`

#### 20.80.3 Typedef Documentation

**20.80.3.1 time\_t** `typedef uint16_t time_t`

## 20.80.4 Function Documentation

**20.80.4.1 clock()** `clock_t clock (`  
`void )`

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()`

**20.80.4.2 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()`

## 20.81 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_EEPPOINTER 21`

## 20.81.1 Macro Definition Documentation

**20.81.1.1 TYPEOF\_INT** `#define TYPEOF_INT 1`

**20.81.1.2 TYPEOF\_SHORT** `#define TYPEOF_SHORT 2`

**20.81.1.3 TYPEOF\_CHAR** `#define TYPEOF_CHAR 3`

**20.81.1.4 TYPEOF\_LONG** `#define TYPEOF_LONG 4`

**20.81.1.5 TYPEOF\_FLOAT** `#define TYPEOF_FLOAT 5`

**20.81.1.6 TYPEOF\_FIXED16X16** `#define TYPEOF_FIXED16X16 6`

**20.81.1.7 TYPEOF\_BIT** `#define TYPEOF_BIT 7`

**20.81.1.8 TYPEOF\_BITFIELD** `#define TYPEOF_BITFIELD 8`

**20.81.1.9 TYPEOF\_SBIT** `#define TYPEOF_SBIT 9`

**20.81.1.10 TYPEOF\_SFR** `#define TYPEOF_SFR 10`

**20.81.1.11 TYPEOF\_VOID** `#define TYPEOF_VOID 11`

**20.81.1.12 TYPEOF\_STRUCT** `#define TYPEOF_STRUCT 12`

**20.81.1.13 TYPEOF\_ARRAY** `#define TYPEOF_ARRAY 13`

**20.81.1.14 TYPEOF\_FUNCTION** `#define TYPEOF_FUNCTION 14`

**20.81.1.15 TYPEOF\_POINTER** `#define TYPEOF_POINTER 15`

**20.81.1.16 TYPEOF\_FPOINTER** `#define TYPEOF_FPOINTER 16`

**20.81.1.17 TYPEOF\_CPOINTER** `#define TYPEOF_CPOINTER 17`



**20.81.1.18** **typeof\_GPOINTER** `#define typeof_GPOINTER 18`

**20.81.1.19** **typeof\_PPOINTER** `#define typeof_PPOINTER 19`

**20.81.1.20** **typeof\_IPOINTER** `#define typeof_IPOINTER 20`

**20.81.1.21** **typeof\_EEPPPOINTER** `#define typeof_EEPPPOINTER 21`



## Index

- `_AUD3WAVERAM`
  - `hardware.h`, [212](#)
- `_BIOS`
  - `sms.h`, [370](#)
- `_HRAM`
  - `hardware.h`, [212](#)
- `_IO`
  - `hardware.h`, [212](#)
- `_OAMRAM`
  - `hardware.h`, [212](#)
- `_RAM`
  - `hardware.h`, [212](#)
- `_RAMBANK`
  - `hardware.h`, [212](#)
- `_SCRN0`
  - `hardware.h`, [212](#)
- `_SCRN1`
  - `hardware.h`, [212](#)
- `_SRAM`
  - `hardware.h`, [212](#)
- `_SYSTEM`
  - `hardware.h`, [223](#)
  - `msx.h`, [301](#)
  - `nes.h`, [340](#)
  - `sms.h`, [370](#)
- `_VRAM`
  - `hardware.h`, [211](#)
- `_VRAM8000`
  - `hardware.h`, [212](#)
- `_VRAM8800`
  - `hardware.h`, [212](#)
- `_VRAM9000`
  - `hardware.h`, [212](#)
- `__BYTES`
  - `hardware.h`, [199](#), [218](#), [230](#)
- `__BYTE_REG`
  - `hardware.h`, [199](#), [218](#), [230](#)
- `__GBDK_VERSION`
  - `version.h`, [275](#)
- `__HandleCrash`
  - `crash_handler.h`, [131](#)
- `__PTRDIFF_T_DEFINED`
  - `stddef.h`, [372](#)
- `__READ_VDP_REG`
  - `msx.h`, [282](#)
  - `sms.h`, [350](#)
- `__REG`
  - `hardware.h`, [199](#), [224](#), [226](#), [227](#)
- `__SHADOW_REG`
  - `hardware.h`, [224](#)
- `__SIZE_T_DEFINED`
  - `stddef.h`, [372](#)
  - `types.h`, [110](#), [111](#), [114](#)
- `__WCHAR_T_DEFINED`
  - `stddef.h`, [372](#)
- `__WRITE_VDP_REG`
  - `msx.h`, [282](#)
  - `sms.h`, [350](#)
- `__WRITE_VDP_REG_UNSAFE`
  - `msx.h`, [282](#)
  - `sms.h`, [350](#)
- `__assert`
  - `assert.h`, [116](#)
- `__bool_true_false_are_defined`
  - `stdbool.h`, [372](#)
- `__call_banked`
  - `far_ptr.h`, [269](#)
- `__call_banked_addr`
  - `far_ptr.h`, [270](#)
- `__call_banked_bank`
  - `far_ptr.h`, [270](#)
- `__call_banked_ptr`
  - `far_ptr.h`, [270](#)
- `__current_base_prop`
  - `metasprites.h`, [249](#), [258](#)
- `__current_base_tile`
  - `metasprites.h`, [249](#), [252](#), [258](#), [263](#)
- `__current_metasprite`
  - `metasprites.h`, [249](#), [252](#), [258](#), [263](#)
- `__far_ptr`, [88](#)
  - `fn`, [89](#)
  - `ofs`, [89](#)
  - `ptr`, [89](#)
  - `seg`, [89](#)
  - `segfn`, [89](#)
  - `segofs`, [89](#)
- `__memcpy`
  - `string.h`, [99](#)
- `__rand_seed`
  - `rand.h`, [343](#)
- `__render_shadow_OAM`
  - `metasprites.h`, [250](#), [252](#), [258](#), [263](#)
- `__setjmp`
  - `setjmp.h`, [344](#)
- `__cpu`
  - `gb.h`, [189](#)
- `__current_1bpp_colors`
  - `gb.h`, [190](#)
  - `msx.h`, [302](#)
  - `nes.h`, [341](#)
  - `sms.h`, [370](#)
- `__current_2bpp_palette`
  - `msx.h`, [302](#)
  - `sms.h`, [370](#)
- `__current_bank`
  - `gb.h`, [190](#)
  - `msx.h`, [301](#)
  - `nes.h`, [340](#)
  - `sms.h`, [353](#)
- `__fixed`, [89](#)

- b, [90](#)
  - h, [89](#)
  - l, [89](#)
  - w, [90](#)
- \_io\_in
  - gb.h, [189](#)
- \_io\_out
  - gb.h, [189](#)
- \_io\_status
  - gb.h, [189](#)
- \_is\_GBA
  - gb.h, [189](#)
- \_map\_tile\_offset
  - gb.h, [190](#)
  - msx.h, [302](#)
  - nes.h, [341](#)
  - sms.h, [370](#)
- \_shadow\_OAM\_OFF
  - msx.h, [302](#)
  - sms.h, [371](#)
- \_shadow\_OAM\_base
  - gb.h, [190](#)
  - msx.h, [302](#)
  - nes.h, [341](#)
  - sms.h, [371](#)
- \_sprites\_OFF
  - sms.h, [371](#)
- \_submap\_tile\_offset
  - gb.h, [190](#)
  - msx.h, [302](#)
  - nes.h, [341](#)
  - sms.h, [370](#)
- \_switch\_prg0
  - nes.h, [340](#)
- abs
  - stdlib.h, [381](#)
- add\_JOY
  - gb.h, [158](#)
  - msx.h, [289](#)
  - sms.h, [358](#)
- add\_LCD
  - gb.h, [156](#)
  - msx.h, [289](#)
  - nes.h, [316](#)
  - sms.h, [357](#)
- add\_low\_priority\_TIM
  - gb.h, [157](#)
- add\_SIO
  - gb.h, [157](#)
  - msx.h, [289](#)
  - sms.h, [357](#)
- add\_TIM
  - gb.h, [157](#)
  - msx.h, [289](#)
  - sms.h, [357](#)
- add\_VBL
  - gb.h, [156](#)
  - msx.h, [289](#)
- nes.h, [315](#)
- sms.h, [357](#)
- AND
  - drawing.h, [132](#)
- arand
  - rand.h, [343](#)
- assert
  - assert.h, [116](#)
- assert.h
  - \_\_assert, [116](#)
  - assert, [116](#)
- AT
  - types.h, [113](#)
- atoi
  - stdlib.h, [381](#)
- atol
  - stdlib.h, [381](#)
- atomic\_flag, [90](#)
  - flag, [90](#)
- atomic\_flag\_clear
  - stdatomic.h, [371](#)
- atomic\_flag\_test\_and\_set
  - stdatomic.h, [371](#)
- AUD1SWEEP\_DOWN
  - hardware.h, [201](#)
- AUD1SWEEP\_LENGTH
  - hardware.h, [201](#)
- AUD1SWEEP\_TIME
  - hardware.h, [201](#)
- AUD1SWEEP\_UP
  - hardware.h, [201](#)
- AUD3WAVE
  - hardware.h, [214](#)
- AUD4POLY\_WIDTH\_15BIT
  - hardware.h, [202](#)
- AUD4POLY\_WIDTH\_7BIT
  - hardware.h, [202](#)
- AUDENA\_OFF
  - hardware.h, [203](#)
- AUDENA\_ON
  - hardware.h, [203](#)
- AUDENV\_DOWN
  - hardware.h, [209](#)
- AUDENV\_LENGTH
  - hardware.h, [210](#)
- AUDENV\_UP
  - hardware.h, [209](#)
- AUDENV\_VOL
  - hardware.h, [209](#)
- AUDHIGH\_LENGTH\_OFF
  - hardware.h, [210](#)
- AUDHIGH\_LENGTH\_ON
  - hardware.h, [210](#)
- AUDHIGH\_RESTART
  - hardware.h, [210](#)
- AUDLEN\_DUTY\_12\_5
  - hardware.h, [209](#)
- AUDLEN\_DUTY\_25

- hardware.h, [209](#)
- AUDLEN\_DUTY\_50
  - hardware.h, [209](#)
- AUDLEN\_DUTY\_75
  - hardware.h, [209](#)
- AUDLEN\_LENGTH
  - hardware.h, [209](#)
- AUDTERM\_1\_LEFT
  - hardware.h, [203](#)
- AUDTERM\_1\_RIGHT
  - hardware.h, [203](#)
- AUDTERM\_2\_LEFT
  - hardware.h, [203](#)
- AUDTERM\_2\_RIGHT
  - hardware.h, [203](#)
- AUDTERM\_3\_LEFT
  - hardware.h, [203](#)
- AUDTERM\_3\_RIGHT
  - hardware.h, [203](#)
- AUDTERM\_4\_LEFT
  - hardware.h, [203](#)
- AUDTERM\_4\_RIGHT
  - hardware.h, [203](#)
- AUDVOL\_VIN\_LEFT
  - hardware.h, [202](#)
- AUDVOL\_VIN\_RIGHT
  - hardware.h, [202](#)
- AUDVOL\_VOL\_LEFT
  - hardware.h, [202](#)
- AUDVOL\_VOL\_RIGHT
  - hardware.h, [202](#)
- b
  - \_fixed, [90](#)
  - emu\_debug.h, [139](#)
  - gb.h, [190](#)
  - msx.h, [302](#)
  - sms.h, [370](#)
- BANK
  - gb.h, [149](#)
  - incbin.h, [273](#)
  - msx.h, [285](#)
  - nes.h, [310](#)
  - sms.h, [353](#)
- BANKED
  - types.h, [113](#)
- BANKREF
  - gb.h, [149](#)
  - msx.h, [285](#)
  - nes.h, [311](#)
  - sms.h, [353](#)
- BANKREF\_EXTERN
  - gb.h, [149](#)
  - msx.h, [285](#)
  - nes.h, [311](#)
  - sms.h, [353](#)
- BCD
  - bcd.h, [118](#), [121](#), [123](#)
- bcd.h
  - BCD, [118](#), [121](#), [123](#)
  - bcd2text, [120](#), [122](#), [124](#)
  - bcd\_add, [120](#), [122](#), [123](#)
  - BCD\_HEX, [118](#), [121](#), [123](#)
  - bcd\_sub, [120](#), [122](#), [124](#)
  - MAKE\_BCD, [118](#), [121](#), [123](#)
  - uint2bcd, [118](#), [121](#), [123](#)
- bcd2text
  - bcd.h, [120](#), [122](#), [124](#)
- bcd\_add
  - bcd.h, [120](#), [122](#), [123](#)
- BCD\_HEX
  - bcd.h, [118](#), [121](#), [123](#)
- bcd\_sub
  - bcd.h, [120](#), [122](#), [124](#)
- BCPD\_REG
  - hardware.h, [216](#)
- BCPS\_REG
  - hardware.h, [216](#)
- BCPSF\_AUTOINC
  - hardware.h, [208](#)
- BGB\_BREAKPOINT
  - emu\_debug.h, [138](#)
- BGB\_MESSAGE
  - emu\_debug.h, [137](#)
- BGB\_printf
  - emu\_debug.h, [138](#)
- BGB\_PROFILE\_BEGIN
  - emu\_debug.h, [137](#)
- BGB\_PROFILE\_END
  - emu\_debug.h, [137](#)
- BGB\_profiler\_message
  - emu\_debug.h, [138](#)
- BGB\_TEXT
  - emu\_debug.h, [138](#)
- BGP\_REG
  - hardware.h, [215](#)
- bkg\_scroll\_x
  - hardware.h, [227](#)
- bkg\_scroll\_y
  - hardware.h, [227](#)
- BKGF\_BANK0
  - hardware.h, [207](#)
- BKGF\_BANK1
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL0
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL1
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL2
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL3
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL4
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL5
  - hardware.h, [207](#)
- BKGF\_CGB\_PAL6

- hardware.h, 207
- BKGF\_CGB\_PAL7
  - hardware.h, 207
- BKGF\_PRI
  - hardware.h, 207
- BKGF\_XFLIP
  - hardware.h, 207
- BKGF\_YFLIP
  - hardware.h, 207
- BLACK
  - drawing.h, 132
- bool
  - stdbool.h, 372
- BOOLEAN
  - types.h, 113
- box
  - drawing.h, 134
- BP\_SIZE
  - setjmp.h, 344
- BPX\_SIZE
  - setjmp.h, 344
- bsearch
  - stdlib.h, 383
- BYTE
  - types.h, 113
- c
  - emu\_debug.h, 139
  - gb.h, 189
  - gbdecompress.h, 193
  - msx.h, 301
  - sgb.h, 266
  - sms.h, 370
  - string.h, 106
- calloc
  - stdlib.h, 383
- cancel\_pending\_interrupts
  - gb.h, 158
  - msx.h, 290
  - sms.h, 358
- cgb.h
  - cgb\_compatibility, 130
  - cpu\_fast, 130
  - cpu\_slow, 130
  - palette\_color\_t, 128
  - RGB, 126
  - RGB8, 126
  - RGB\_AQUA, 127
  - RGB\_BLACK, 127
  - RGB\_BLUE, 127
  - RGB\_BROWN, 128
  - RGB\_CYAN, 127
  - RGB\_DARKBLUE, 127
  - RGB\_DARKGRAY, 127
  - RGB\_DARKGREEN, 127
  - RGB\_DARKRED, 127
  - RGB\_DARKYELLOW, 127
  - RGB\_GREEN, 127
  - RGB\_LIGHTFLESH, 127
  - RGB\_LIGHTGRAY, 127
  - RGB\_ORANGE, 128
  - RGB\_PINK, 127
  - RGB\_PURPLE, 127
  - RGB\_RED, 127
  - RGB\_TEAL, 128
  - RGB\_WHITE, 127
  - RGB\_YELLOW, 127
  - RGBHTML, 126
  - set\_bkg\_palette, 128
  - set\_bkg\_palette\_entry, 129
  - set\_default\_palette, 130
  - set\_sprite\_palette, 128
  - set\_sprite\_palette\_entry, 129
- cgb\_compatibility
  - cgb.h, 130
  - sms.h, 360
- CGB\_TYPE
  - gb.h, 148
- CHAR\_BIT
  - limits.h, 275
- CHAR\_MAX
  - limits.h, 276
- CHAR\_MIN
  - limits.h, 276
- circle
  - drawing.h, 134
- clock
  - time.h, 385
- clock\_t
  - types.h, 111, 112, 115
- CLOCKS\_PER\_SEC
  - time.h, 384
- cls
  - console.h, 267
- color
  - drawing.h, 135
- COMPAT\_PALETTE
  - gb.h, 154
  - msx.h, 287
  - nes.h, 313
  - sms.h, 355
- console.h
  - cls, 267
  - gotoxy, 266
  - posx, 266
  - posy, 266
  - setchar, 267
- cpu\_fast
  - cgb.h, 130
  - msx.h, 292
  - sms.h, 360
- cpu\_slow
  - cgb.h, 130
- crash\_handler.h
  - \_\_HandleCrash, 131
- CRITICAL
  - types.h, 113

ctype.h  
  isalpha, 116  
  isdigit, 117  
  islower, 117  
  isspace, 117  
  isupper, 117  
  tolower, 117  
  toupper, 117  
CURRENT\_BANK  
  gb.h, 149  
  msx.h, 285  
  nes.h, 310  
  sms.h, 353  
  
d  
  gb.h, 189  
  msx.h, 301  
  sms.h, 370  
delay  
  gb.h, 159  
  msx.h, 291  
  nes.h, 316  
  sms.h, 358  
DEVICE\_SCREEN\_BUFFER\_HEIGHT  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_BUFFER\_WIDTH  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_HEIGHT  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_MAP\_ENTRY\_SIZE  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_PX\_HEIGHT  
  hardware.h, 211, 222, 226, 239  
DEVICE\_SCREEN\_PX\_WIDTH  
  hardware.h, 211, 222, 226, 239  
DEVICE\_SCREEN\_WIDTH  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_X\_OFFSET  
  hardware.h, 211, 225  
DEVICE\_SCREEN\_Y\_OFFSET  
  hardware.h, 211, 225  
DEVICE\_SPRITE\_PX\_OFFSET\_X  
  hardware.h, 211, 226  
DEVICE\_SPRITE\_PX\_OFFSET\_Y  
  hardware.h, 211, 226  
DEVICE\_SUPPORTS\_COLOR  
  gb.h, 148  
  msx.h, 285  
  sms.h, 352  
DEVICE\_WINDOW\_PX\_OFFSET\_X  
  hardware.h, 211, 226  
DEVICE\_WINDOW\_PX\_OFFSET\_Y  
  hardware.h, 211, 226  
disable\_interrupts  
  gb.h, 161  
  msx.h, 292  
  nes.h, 319  
  sms.h, 360  
DISABLE\_OAM\_DMA  
  gb.h, 154  
  nes.h, 313  
DISABLE\_RAM  
  gb.h, 150  
  msx.h, 286  
  nes.h, 312  
  sms.h, 354  
DISABLE\_RAM\_MBC1  
  gb.h, 151  
DISABLE\_RAM\_MBC5  
  gb.h, 152  
DISABLE\_VBL\_TRANSFER  
  gb.h, 154  
  msx.h, 287  
  nes.h, 314  
  sms.h, 355  
DISPLAY\_OFF  
  gb.h, 153  
  msx.h, 284  
  nes.h, 312  
  sms.h, 351  
display\_off  
  gb.h, 162  
  msx.h, 290  
  nes.h, 319  
  sms.h, 358  
DISPLAY\_ON  
  gb.h, 152  
  msx.h, 283  
  nes.h, 312  
  sms.h, 351  
display\_on  
  nes.h, 319  
DIV\_REG  
  hardware.h, 213  
  msx.h, 285  
  sms.h, 353  
DKGREY  
  drawing.h, 132  
DMA\_REG  
  hardware.h, 215  
DMG\_BLACK  
  gb.h, 146  
  nes.h, 309  
DMG\_DARK\_GRAY  
  gb.h, 146  
  nes.h, 309  
DMG\_LITE\_GRAY  
  gb.h, 147  
  nes.h, 310  
DMG\_PALETTE  
  gb.h, 147  
  nes.h, 310  
DMG\_TYPE  
  gb.h, 148  
DMG\_WHITE  
  gb.h, 147  
  nes.h, 310

[docs/pages/01\\_getting\\_started.md](#), 95  
[docs/pages/02\\_links\\_and\\_tools.md](#), 95  
[docs/pages/03\\_using\\_gbdk.md](#), 95  
[docs/pages/04\\_coding\\_guidelines.md](#), 95  
[docs/pages/05\\_banking\\_mbc5.md](#), 95  
[docs/pages/06\\_toolchain.md](#), 95  
[docs/pages/06b\\_supported\\_consoles.md](#), 95  
[docs/pages/07\\_sample\\_programs.md](#), 95  
[docs/pages/08\\_faq.md](#), 95  
[docs/pages/09\\_migrating\\_new\\_versions.md](#), 95  
[docs/pages/10\\_release\\_notes.md](#), 95  
[docs/pages/20\\_toolchain\\_settings.md](#), 95  
[docs/pages/docs\\_index.md](#), 95  
[draw\\_image](#)  
     [drawing.h](#), 134  
[drawing.h](#)  
     AND, 132  
     BLACK, 132  
     box, 134  
     circle, 134  
     color, 135  
     DKGREY, 132  
     draw\_image, 134  
     getpix, 135  
     gotogxy, 135  
     gprint, 133  
     gprintf, 133  
     gprintln, 133  
     gprintn, 133  
     GRAPHICS\_HEIGHT, 132  
     GRAPHICS\_WIDTH, 132  
     line, 134  
     LTGREY, 132  
     M\_FILL, 132  
     M\_NOFILL, 132  
     OR, 132  
     plot, 134  
     plot\_point, 134  
     SIGNED, 132  
     SOLID, 132  
     switch\_data, 134  
     UNSIGNED, 133  
     WHITE, 132  
     wrtchr, 135  
     XOR, 132  
[dtile](#)  
     metasprite\_t, 93  
[DWORD](#)  
     types.h, 113  
[dx](#)  
     metasprite\_t, 93  
[dy](#)  
     metasprite\_t, 93  
[e](#)  
     gb.h, 189  
     msx.h, 301  
     sms.h, 370  
[EMPTY\\_IFLAG](#)  
     gb.h, 146  
     msx.h, 282  
     sms.h, 350  
[EMU\\_BREAKPOINT](#)  
     emu\_debug.h, 138  
[emu\\_debug.h](#)  
     b, 139  
     BGB\_BREAKPOINT, 138  
     BGB\_MESSAGE, 137  
     BGB\_printf, 138  
     BGB\_PROFILE\_BEGIN, 137  
     BGB\_PROFILE\_END, 137  
     BGB\_profiler\_message, 138  
     BGB\_TEXT, 138  
     c, 139  
     EMU\_BREAKPOINT, 138  
     EMU\_fmtbuf, 138  
     EMU\_MESSAGE, 136  
     EMU\_printf, 138  
     EMU\_PROFILE\_BEGIN, 137  
     EMU\_PROFILE\_END, 137  
     EMU\_profiler\_message, 138  
     EMU\_TEXT, 137  
[EMU\\_fmtbuf](#)  
     emu\_debug.h, 138  
[EMU\\_MESSAGE](#)  
     emu\_debug.h, 136  
[EMU\\_printf](#)  
     emu\_debug.h, 138  
[EMU\\_PROFILE\\_BEGIN](#)  
     emu\_debug.h, 137  
[EMU\\_PROFILE\\_END](#)  
     emu\_debug.h, 137  
[EMU\\_profiler\\_message](#)  
     emu\_debug.h, 138  
[EMU\\_TEXT](#)  
     emu\_debug.h, 137  
[enable\\_interrupts](#)  
     gb.h, 161  
     msx.h, 291  
     nes.h, 318  
     sms.h, 359  
[ENABLE\\_OAM\\_DMA](#)  
     gb.h, 154  
     nes.h, 314  
[ENABLE\\_RAM](#)  
     gb.h, 150  
     msx.h, 286  
     nes.h, 312  
     sms.h, 354  
[ENABLE\\_RAM\\_MBC1](#)  
     gb.h, 151  
[ENABLE\\_RAM\\_MBC5](#)  
     gb.h, 152  
[ENABLE\\_VBL\\_TRANSFER](#)  
     gb.h, 154  
     msx.h, 287  
     nes.h, 314



- sms.h, [355](#)
- exit
  - stdlib.h, [381](#)
- FALSE
  - types.h, [115](#)
- false
  - stdbool.h, [372](#)
- FAR\_CALL
  - far\_ptr.h, [269](#)
- FAR\_FUNC
  - far\_ptr.h, [268](#)
- FAR\_OFS
  - far\_ptr.h, [268](#)
- FAR\_PTR
  - far\_ptr.h, [269](#)
- far\_ptr.h
  - \_\_call\_banked, [269](#)
  - \_\_call\_banked\_addr, [270](#)
  - \_\_call\_banked\_bank, [270](#)
  - \_\_call\_banked\_ptr, [270](#)
  - FAR\_CALL, [269](#)
  - FAR\_FUNC, [268](#)
  - FAR\_OFS, [268](#)
  - FAR\_PTR, [269](#)
  - FAR\_SEG, [268](#)
  - TO\_FAR\_PTR, [268](#)
  - to\_far\_ptr, [269](#)
- FAR\_SEG
  - far\_ptr.h, [268](#)
- fill\_bkg\_rect
  - gb.h, [188](#)
  - msx.h, [287](#)
  - nes.h, [340](#)
  - sms.h, [355](#)
- fill\_rect
  - gb.h, [155](#)
  - msx.h, [296](#)
  - nes.h, [314](#)
  - sms.h, [366](#)
- fill\_rect\_compat
  - sms.h, [366](#)
- fill\_win\_rect
  - gb.h, [188](#)
  - msx.h, [287](#)
  - sms.h, [355](#)
- first\_tile
  - sfont\_handle, [95](#)
- fixed
  - types.h, [114](#)
- flag
  - atomic\_flag, [90](#)
- flush\_shadow\_attributes
  - nes.h, [340](#)
- fn
  - \_\_far\_ptr, [89](#)
- font
  - sfont\_handle, [95](#)
- font.h
  - FONT\_128ENCODING, [271](#)
  - FONT\_256ENCODING, [271](#)
  - font\_color, [272](#)
  - FONT\_COMPRESSED, [271](#)
  - font\_init, [271](#)
  - font\_load, [271](#)
  - FONT\_NOENCODING, [271](#)
  - font\_set, [271](#)
  - font\_t, [271](#)
  - mfont\_handle, [271](#)
  - pmfont\_handle, [271](#)
- FONT\_128ENCODING
  - font.h, [271](#)
- FONT\_256ENCODING
  - font.h, [271](#)
- font\_color
  - font.h, [272](#)
- FONT\_COMPRESSED
  - font.h, [271](#)
- font\_ibm
  - List of gbk fonts, [88](#)
- font\_ibm\_fixed
  - List of gbk fonts, [88](#)
- font\_init
  - font.h, [271](#)
- font\_italic
  - List of gbk fonts, [88](#)
- font\_load
  - font.h, [271](#)
- font\_min
  - List of gbk fonts, [88](#)
- FONT\_NOENCODING
  - font.h, [271](#)
- font\_set
  - font.h, [271](#)
- font\_spect
  - List of gbk fonts, [88](#)
- font\_t
  - font.h, [271](#)
- free
  - stdlib.h, [383](#)
- func
  - isr\_nested\_vector\_t, [90](#)
  - isr\_vector\_t, [91](#)
- GAMEBOY
  - gb.h, [144](#)
- gb.h
  - \_cpu, [189](#)
  - \_current\_1bpp\_colors, [190](#)
  - \_current\_bank, [190](#)
  - \_io\_in, [189](#)
  - \_io\_out, [189](#)
  - \_io\_status, [189](#)
  - \_is\_GBA, [189](#)
  - \_map\_tile\_offset, [190](#)
  - \_shadow\_OAM\_base, [190](#)
  - \_submap\_tile\_offset, [190](#)
  - add\_JOY, [158](#)

add\_LCD, 156  
add\_low\_priority\_TIM, 157  
add\_SIO, 157  
add\_TIM, 157  
add\_VBL, 156  
b, 190  
BANK, 149  
BANKREF, 149  
BANKREF\_EXTERN, 149  
c, 189  
cancel\_pending\_interrupts, 158  
CGB\_TYPE, 148  
COMPAT\_PALETTE, 154  
CURRENT\_BANK, 149  
d, 189  
delay, 159  
DEVICE\_SUPPORTS\_COLOR, 148  
disable\_interrupts, 161  
DISABLE\_OAM\_DMA, 154  
DISABLE\_RAM, 150  
DISABLE\_RAM\_MBC1, 151  
DISABLE\_RAM\_MBC5, 152  
DISABLE\_VBL\_TRANSFER, 154  
DISPLAY\_OFF, 153  
display\_off, 162  
DISPLAY\_ON, 152  
DMG\_BLACK, 146  
DMG\_DARK\_GRAY, 146  
DMG\_LITE\_GRAY, 147  
DMG\_PALETTE, 147  
DMG\_TYPE, 148  
DMG\_WHITE, 147  
e, 189  
EMPTY\_IFLAG, 146  
enable\_interrupts, 161  
ENABLE\_OAM\_DMA, 154  
ENABLE\_RAM, 150  
ENABLE\_RAM\_MBC1, 151  
ENABLE\_RAM\_MBC5, 152  
ENABLE\_VBL\_TRANSFER, 154  
fill\_bkg\_rect, 188  
fill\_rect, 155  
fill\_win\_rect, 188  
GAMEBOY, 144  
GBA\_DETECTED, 148  
GBA\_NOT\_DETECTED, 148  
get\_bkg\_data, 166  
get\_bkg\_tile\_xy, 172  
get\_bkg\_tiles, 171  
get\_bkg\_xy\_addr, 163  
get\_data, 184  
get\_mode, 159  
get\_sprite\_data, 180  
get\_sprite\_prop, 182  
get\_sprite\_tile, 181  
get\_system, 159  
get\_tiles, 185  
get\_vram\_byte, 163  
get\_win\_data, 174  
get\_win\_tile\_xy, 178  
get\_win\_tiles, 177  
get\_win\_xy\_addr, 173  
h, 189  
HARDWARE\_SPRITE\_CAN\_FLIP\_X, 154  
HARDWARE\_SPRITE\_CAN\_FLIP\_Y, 155  
HIDE\_BKG, 153  
HIDE\_LEFT\_COLUMN, 153  
hide\_sprite, 183  
HIDE\_SPRITES, 153  
HIDE\_WIN, 153  
hiramcpy, 162  
init\_bkg, 187  
init\_win, 187  
int\_handler, 155  
IO\_ERROR, 149  
IO\_IDLE, 149  
IO\_RECEIVING, 149  
IO\_SENDING, 149  
J\_A, 144  
J\_B, 144  
J\_DOWN, 144  
J\_LEFT, 144  
J\_RIGHT, 144  
J\_SELECT, 144  
J\_START, 144  
J\_UP, 144  
JOY\_IFLAG, 146  
joypad, 159  
joypad\_ex, 161  
joypad\_init, 160  
l, 189  
LCD\_IFLAG, 146  
M\_DRAWING, 144  
M\_NO\_INTERP, 145  
M\_NO\_SCROLL, 144  
M\_TEXT\_INOUT, 144  
M\_TEXT\_OUT, 144  
MAX\_HARDWARE\_SPRITES, 154  
MAXWNDPOSX, 147  
MAXWNDPOSY, 148  
MGB\_TYPE, 148  
MINWNDPOSX, 147  
MINWNDPOSY, 147  
mode, 158  
move\_bkg, 173  
move\_sprite, 182  
move\_win, 179  
NINTENDO, 143  
nowait\_int\_handler, 158  
OAM\_item\_t, 155  
receive\_byte, 159  
refresh\_OAM, 162  
remove\_JOY, 156  
remove\_LCD, 155  
remove\_SIO, 156  
remove\_TIM, 155

remove\_VBL, 155  
reset, 162  
S\_BANK, 145  
S\_FLIPX, 145  
S\_FLIPY, 145  
S\_PAL, 145  
S\_PALETTE, 145  
S\_PRIORITY, 145  
SCREENHEIGHT, 147  
SCREENWIDTH, 147  
scroll\_bkg, 173  
scroll\_sprite, 183  
scroll\_win, 179  
send\_byte, 159  
set\_1bpp\_colors, 163  
set\_1bpp\_colors\_ex, 163  
set\_2bpp\_palette, 163  
set\_attribute\_xy, 154  
set\_bkg\_1bpp\_data, 164  
set\_bkg\_2bpp\_data, 154  
set\_bkg\_attribute\_xy, 172  
set\_bkg\_attributes, 168  
set\_bkg\_based\_submap, 170  
set\_bkg\_based\_tiles, 167  
set\_bkg\_data, 164  
set\_bkg\_native\_data, 186  
set\_bkg\_submap, 169  
set\_bkg\_submap\_attributes, 170  
set\_bkg\_tile\_xy, 172  
set\_bkg\_tiles, 166  
SET\_BORDER\_COLOR, 153  
set\_data, 183  
set\_interrupts, 161  
set\_native\_tile\_data, 186  
SET\_SHADOW\_OAM\_ADDRESS, 180  
set\_sprite\_1bpp\_data, 180  
set\_sprite\_2bpp\_data, 154  
set\_sprite\_data, 179  
set\_sprite\_native\_data, 187  
set\_sprite\_prop, 181  
set\_sprite\_tile, 181  
set\_tile\_data, 185  
set\_tile\_map, 154  
set\_tile\_submap, 154  
set\_tile\_xy, 154  
set\_tiles, 184  
set\_vram\_byte, 163  
set\_win\_1bpp\_data, 174  
set\_win\_based\_submap, 177  
set\_win\_based\_tiles, 175  
set\_win\_data, 173  
set\_win\_submap, 176  
set\_win\_tile\_xy, 178  
set\_win\_tiles, 175  
shadow\_OAM, 190  
SHOW\_BKG, 153  
SHOW\_LEFT\_COLUMN, 153  
SHOW\_SPRITES, 153  
SHOW\_WIN, 153  
SIO\_IFLAG, 146  
SPRITES\_8x16, 154  
SPRITES\_8x8, 154  
SWITCH\_16\_8\_MODE\_MBC1, 151  
SWITCH\_4\_32\_MODE\_MBC1, 151  
SWITCH\_RAM, 150  
SWITCH\_RAM\_MBC1, 151  
SWITCH\_RAM\_MBC5, 152  
SWITCH\_ROM, 150  
SWITCH\_ROM\_MBC1, 151  
SWITCH\_ROM\_MBC5, 151  
SWITCH\_ROM\_MBC5\_8M, 152  
SWITCH\_ROM\_MEGADUCK, 151  
sys\_time, 189  
SYSTEM\_50HZ, 144  
SYSTEM\_60HZ, 143  
TIM\_IFLAG, 146  
VBL\_IFLAG, 146  
vmemcpy, 184  
vmemset, 188  
vsync, 162  
wait\_int\_handler, 158  
wait\_vbl\_done, 162  
waitpad, 160  
waitpadup, 160  
gb\_decompress  
    gbdecompress.h, 190, 192  
gb\_decompress\_bkg\_data  
    gbdecompress.h, 191  
gb\_decompress\_sprite\_data  
    gbdecompress.h, 192  
gb\_decompress\_win\_data  
    gbdecompress.h, 191  
GBA\_DETECTED  
    gb.h, 148  
GBA\_NOT\_DETECTED  
    gb.h, 148  
gbdecompress.h  
    c, 193  
    gb\_decompress, 190, 192  
    gb\_decompress\_bkg\_data, 191  
    gb\_decompress\_sprite\_data, 192  
    gb\_decompress\_win\_data, 191  
gbdk-lib/include/asm/mos6502/provides.h, 95  
gbdk-lib/include/asm/mos6502/stdarg.h, 96  
gbdk-lib/include/asm/mos6502/string.h, 98  
gbdk-lib/include/asm/mos6502/types.h, 110  
gbdk-lib/include/asm/sm83/provides.h, 96  
gbdk-lib/include/asm/sm83/stdarg.h, 97  
gbdk-lib/include/asm/sm83/string.h, 102  
gbdk-lib/include/asm/sm83/types.h, 111  
gbdk-lib/include/asm/types.h, 112  
gbdk-lib/include/asm/z80/provides.h, 96  
gbdk-lib/include/asm/z80/stdarg.h, 98  
gbdk-lib/include/asm/z80/string.h, 106  
gbdk-lib/include/asm/z80/types.h, 114  
gbdk-lib/include/assert.h, 116

- gbdk-lib/include/ctype.h, 116
- gbdk-lib/include/gb/bcd.h, 118
- gbdk-lib/include/gb/bgb\_emu.h, 124
- gbdk-lib/include/gb/cgb.h, 124
- gbdk-lib/include/gb/crash\_handler.h, 130
- gbdk-lib/include/gb/drawing.h, 131
- gbdk-lib/include/gb/emu\_debug.h, 135
- gbdk-lib/include/gb/gb.h, 139
- gbdk-lib/include/gb/gbdecompress.h, 190
- gbdk-lib/include/gb/hardware.h, 193
- gbdk-lib/include/gb/hblankcpy.h, 240
- gbdk-lib/include/gb/isr.h, 241
- gbdk-lib/include/gb/metasprites.h, 243
- gbdk-lib/include/gb/sgb.h, 263
- gbdk-lib/include/gbdk/bcd.h, 120
- gbdk-lib/include/gbdk/console.h, 266
- gbdk-lib/include/gbdk/emu\_debug.h, 136
- gbdk-lib/include/gbdk/far\_ptr.h, 267
- gbdk-lib/include/gbdk/font.h, 270
- gbdk-lib/include/gbdk/gbdecompress.h, 192
- gbdk-lib/include/gbdk/gbdk-lib.h, 272
- gbdk-lib/include/gbdk/incbin.h, 272
- gbdk-lib/include/gbdk/metasprites.h, 250
- gbdk-lib/include/gbdk/platform.h, 274
- gbdk-lib/include/gbdk/riledecompress.h, 274
- gbdk-lib/include/gbdk/version.h, 275
- gbdk-lib/include/limits.h, 275
- gbdk-lib/include/msx/hardware.h, 216
- gbdk-lib/include/msx/metasprites.h, 250
- gbdk-lib/include/msx/msx.h, 277
- gbdk-lib/include/nes/bcd.h, 121
- gbdk-lib/include/nes/hardware.h, 223
- gbdk-lib/include/nes/metasprites.h, 253
- gbdk-lib/include/nes/nes.h, 303
- gbdk-lib/include/nes/rgb\_to\_nes\_macro.h, 341
- gbdk-lib/include/rand.h, 341
- gbdk-lib/include/setjmp.h, 343
- gbdk-lib/include/sms/bcd.h, 122
- gbdk-lib/include/sms/gbdecompress.h, 192
- gbdk-lib/include/sms/hardware.h, 227
- gbdk-lib/include/sms/metasprites.h, 258
- gbdk-lib/include/sms/sms.h, 344
- gbdk-lib/include/stdarg.h, 98
- gbdk-lib/include/stdatomic.h, 371
- gbdk-lib/include/stdbool.h, 372
- gbdk-lib/include/stddef.h, 372
- gbdk-lib/include/stdint.h, 373
- gbdk-lib/include/stdio.h, 379
- gbdk-lib/include/stdlib.h, 380
- gbdk-lib/include/stdnoreturn.h, 384
- gbdk-lib/include/string.h, 110
- gbdk-lib/include/time.h, 384
- gbdk-lib/include/typeof.h, 385
- gbdk-lib/include/types.h, 115
- get\_bkg\_data
  - gb.h, 166
- get\_bkg\_tile\_xy
  - gb.h, 172
- nes.h, 329
- get\_bkg\_tiles
  - gb.h, 171
  - nes.h, 327
- get\_bkg\_xy\_addr
  - gb.h, 163
  - msx.h, 301
  - nes.h, 320
  - sms.h, 369
- get\_data
  - gb.h, 184
- get\_mode
  - gb.h, 159
  - msx.h, 288
  - nes.h, 316
  - sms.h, 356
- get\_r\_reg
  - msx.h, 290
  - sms.h, 358
- get\_sprite\_data
  - gb.h, 180
- get\_sprite\_prop
  - gb.h, 182
  - msx.h, 299
  - nes.h, 334
  - sms.h, 367
- get\_sprite\_tile
  - gb.h, 181
  - msx.h, 297
  - nes.h, 332
  - sms.h, 367
- get\_system
  - gb.h, 159
  - msx.h, 288
  - nes.h, 316
  - sms.h, 356
- get\_tiles
  - gb.h, 185
- get\_vram\_byte
  - gb.h, 163
- get\_win\_data
  - gb.h, 174
- get\_win\_tile\_xy
  - gb.h, 178
- get\_win\_tiles
  - gb.h, 177
- get\_win\_xy\_addr
  - gb.h, 173
  - msx.h, 287
  - sms.h, 356
- getchar
  - stdio.h, 380
- getpix
  - drawing.h, 135
- gets
  - stdio.h, 380
- GGEXT\_NINIT
  - hardware.h, 231

- GGSTATE\_NJAP
  - hardware.h, [230](#)
- GGSTATE\_NNTS
  - hardware.h, [231](#)
- GGSTATE\_STT
  - hardware.h, [230](#)
- gotogxy
  - drawing.h, [135](#)
- gotoxy
  - console.h, [266](#)
- gprint
  - drawing.h, [133](#)
- gprintf
  - drawing.h, [133](#)
- gprintln
  - drawing.h, [133](#)
- gprintrn
  - drawing.h, [133](#)
- GRAPHICS\_HEIGHT
  - drawing.h, [132](#)
- GRAPHICS\_WIDTH
  - drawing.h, [132](#)
- GUN\_P1\_LATCH
  - hardware.h, [232](#)
- GUN\_P2\_LATCH
  - hardware.h, [233](#)
- h
  - \_fixed, [89](#)
  - gb.h, [189](#)
  - msx.h, [301](#)
  - sms.h, [370](#)
- hardware.h
  - \_AUD3WAVERAM, [212](#)
  - \_HRAM, [212](#)
  - \_IO, [212](#)
  - \_OAMRAM, [212](#)
  - \_RAM, [212](#)
  - \_RAMBANK, [212](#)
  - \_SCRN0, [212](#)
  - \_SCRN1, [212](#)
  - \_SRAM, [212](#)
  - \_SYSTEM, [223](#)
  - \_VRAM, [211](#)
  - \_VRAM8000, [212](#)
  - \_VRAM8800, [212](#)
  - \_VRAM9000, [212](#)
  - \_\_BYTES, [199](#), [218](#), [230](#)
  - \_\_BYTE\_REG, [199](#), [218](#), [230](#)
  - \_\_REG, [199](#), [224](#), [226](#), [227](#)
  - \_\_SHADOW\_REG, [224](#)
  - AUD1SWEEP\_DOWN, [201](#)
  - AUD1SWEEP\_LENGTH, [201](#)
  - AUD1SWEEP\_TIME, [201](#)
  - AUD1SWEEP\_UP, [201](#)
  - AUD3WAVE, [214](#)
  - AUD4POLY\_WIDTH\_15BIT, [202](#)
  - AUD4POLY\_WIDTH\_7BIT, [202](#)
  - AUDENA\_OFF, [203](#)
  - AUDENA\_ON, [203](#)
  - AUDENV\_DOWN, [209](#)
  - AUDENV\_LENGTH, [210](#)
  - AUDENV\_UP, [209](#)
  - AUDENV\_VOL, [209](#)
  - AUDHIGH\_LENGTH\_OFF, [210](#)
  - AUDHIGH\_LENGTH\_ON, [210](#)
  - AUDHIGH\_RESTART, [210](#)
  - AUDLEN\_DUTY\_12\_5, [209](#)
  - AUDLEN\_DUTY\_25, [209](#)
  - AUDLEN\_DUTY\_50, [209](#)
  - AUDLEN\_DUTY\_75, [209](#)
  - AUDLEN\_LENGTH, [209](#)
  - AUDTERM\_1\_LEFT, [203](#)
  - AUDTERM\_1\_RIGHT, [203](#)
  - AUDTERM\_2\_LEFT, [203](#)
  - AUDTERM\_2\_RIGHT, [203](#)
  - AUDTERM\_3\_LEFT, [203](#)
  - AUDTERM\_3\_RIGHT, [203](#)
  - AUDTERM\_4\_LEFT, [203](#)
  - AUDTERM\_4\_RIGHT, [203](#)
  - AUDVOL\_VIN\_LEFT, [202](#)
  - AUDVOL\_VIN\_RIGHT, [202](#)
  - AUDVOL\_VOL\_LEFT, [202](#)
  - AUDVOL\_VOL\_RIGHT, [202](#)
  - BCPD\_REG, [216](#)
  - BCPS\_REG, [216](#)
  - BCPSF\_AUTOINC, [208](#)
  - BGP\_REG, [215](#)
  - bkg\_scroll\_x, [227](#)
  - bkg\_scroll\_y, [227](#)
  - BKGF\_BANK0, [207](#)
  - BKGF\_BANK1, [207](#)
  - BKGF\_CGB\_PAL0, [207](#)
  - BKGF\_CGB\_PAL1, [207](#)
  - BKGF\_CGB\_PAL2, [207](#)
  - BKGF\_CGB\_PAL3, [207](#)
  - BKGF\_CGB\_PAL4, [207](#)
  - BKGF\_CGB\_PAL5, [207](#)
  - BKGF\_CGB\_PAL6, [207](#)
  - BKGF\_CGB\_PAL7, [207](#)
  - BKGF\_PRI, [207](#)
  - BKGF\_XFLIP, [207](#)
  - BKGF\_YFLIP, [207](#)
  - DEVICE\_SCREEN\_BUFFER\_HEIGHT, [211](#), [225](#)
  - DEVICE\_SCREEN\_BUFFER\_WIDTH, [211](#), [225](#)
  - DEVICE\_SCREEN\_HEIGHT, [211](#), [225](#)
  - DEVICE\_SCREEN\_MAP\_ENTRY\_SIZE, [211](#), [225](#)
  - DEVICE\_SCREEN\_PX\_HEIGHT, [211](#), [222](#), [226](#), [239](#)
  - DEVICE\_SCREEN\_PX\_WIDTH, [211](#), [222](#), [226](#), [239](#)
  - DEVICE\_SCREEN\_WIDTH, [211](#), [225](#)
  - DEVICE\_SCREEN\_X\_OFFSET, [211](#), [225](#)
  - DEVICE\_SCREEN\_Y\_OFFSET, [211](#), [225](#)
  - DEVICE\_SPRITE\_PX\_OFFSET\_X, [211](#), [226](#)
  - DEVICE\_SPRITE\_PX\_OFFSET\_Y, [211](#), [226](#)
  - DEVICE\_WINDOW\_PX\_OFFSET\_X, [211](#), [226](#)

DEVICE\_WINDOW\_PX\_OFFSET\_Y, [211](#), [226](#)  
DIV\_REG, [213](#)  
DMA\_REG, [215](#)  
GGEXT\_NINIT, [231](#)  
GGSTATE\_NJAP, [230](#)  
GGSTATE\_NNTS, [231](#)  
GGSTATE\_STT, [230](#)  
GUN\_P1\_LATCH, [232](#)  
GUN\_P2\_LATCH, [233](#)  
HDMA1\_REG, [215](#)  
HDMA2\_REG, [215](#)  
HDMA3\_REG, [215](#)  
HDMA4\_REG, [216](#)  
HDMA5\_REG, [216](#)  
HDMA5F\_BUSY, [208](#)  
HDMA5F\_MODE\_GP, [208](#)  
HDMA5F\_MODE\_HBL, [208](#)  
IE\_REG, [216](#)  
IEF\_HILO, [209](#)  
IEF\_SERIAL, [209](#)  
IEF\_STAT, [209](#)  
IEF\_TIMER, [209](#)  
IEF\_VBLANK, [209](#)  
IF\_REG, [213](#)  
JOY\_P1\_DOWN, [237](#)  
JOY\_P1\_LEFT, [237](#)  
JOY\_P1\_LIGHT, [238](#)  
JOY\_P1\_MD\_A, [237](#)  
JOY\_P1\_MD\_MODE, [237](#)  
JOY\_P1\_MD\_START, [237](#)  
JOY\_P1\_MD\_X, [237](#)  
JOY\_P1\_MD\_Y, [237](#)  
JOY\_P1\_MD\_Z, [237](#)  
JOY\_P1\_RIGHT, [237](#)  
JOY\_P1\_SW1, [237](#)  
JOY\_P1\_SW2, [237](#)  
JOY\_P1\_TH\_DIR\_IN, [232](#)  
JOY\_P1\_TH\_DIR\_OUT, [232](#)  
JOY\_P1\_TH\_OUT\_HI, [233](#)  
JOY\_P1\_TH\_OUT\_LO, [233](#)  
JOY\_P1\_TR\_DIR\_IN, [232](#)  
JOY\_P1\_TR\_DIR\_OUT, [232](#)  
JOY\_P1\_TR\_OUT\_HI, [233](#)  
JOY\_P1\_TR\_OUT\_LO, [233](#)  
JOY\_P1\_TRIGGER, [237](#)  
JOY\_P1\_UP, [237](#)  
JOY\_P2\_DOWN, [238](#)  
JOY\_P2\_LEFT, [238](#)  
JOY\_P2\_LIGHT, [238](#)  
JOY\_P2\_MD\_A, [238](#)  
JOY\_P2\_MD\_MODE, [238](#)  
JOY\_P2\_MD\_START, [238](#)  
JOY\_P2\_MD\_X, [238](#)  
JOY\_P2\_MD\_Y, [238](#)  
JOY\_P2\_MD\_Z, [238](#)  
JOY\_P2\_RIGHT, [238](#)  
JOY\_P2\_SW1, [238](#)  
JOY\_P2\_SW2, [238](#)  
JOY\_P2\_TH\_DIR\_IN, [233](#)  
JOY\_P2\_TH\_DIR\_OUT, [233](#)  
JOY\_P2\_TH\_OUT\_HI, [233](#)  
JOY\_P2\_TH\_OUT\_LO, [233](#)  
JOY\_P2\_TR\_DIR\_IN, [232](#)  
JOY\_P2\_TR\_DIR\_OUT, [233](#)  
JOY\_P2\_TR\_OUT\_HI, [233](#)  
JOY\_P2\_TR\_OUT\_LO, [233](#)  
JOY\_P2\_TRIGGER, [238](#)  
JOY\_P2\_UP, [238](#)  
JOY\_RESET, [238](#)  
JOY\_TH\_HI, [233](#)  
JOY\_TH\_LO, [233](#)  
KEY1\_REG, [215](#)  
KEY1F\_DBLSPEED, [206](#)  
KEY1F\_PREPARE, [206](#)  
LCDDC\_REG, [214](#)  
LCDCF\_B\_BG8000, [204](#)  
LCDCF\_B\_BG9C00, [204](#)  
LCDCF\_B\_BGON, [204](#)  
LCDCF\_B\_OBJ16, [204](#)  
LCDCF\_B\_OBJON, [204](#)  
LCDCF\_B\_ON, [204](#)  
LCDCF\_B\_WIN9C00, [204](#)  
LCDCF\_B\_WINON, [204](#)  
LCDCF\_BG8000, [204](#)  
LCDCF\_BG8800, [204](#)  
LCDCF\_BG9800, [204](#)  
LCDCF\_BG9C00, [204](#)  
LCDCF\_BGOFF, [204](#)  
LCDCF\_BGON, [204](#)  
LCDCF\_OBJ16, [204](#)  
LCDCF\_OBJ8, [204](#)  
LCDCF\_OBJOFF, [204](#)  
LCDCF\_OBJON, [204](#)  
LCDCF\_OFF, [203](#)  
LCDCF\_ON, [203](#)  
LCDCF\_WIN9800, [203](#)  
LCDCF\_WIN9C00, [203](#)  
LCDCF\_WINOFF, [203](#)  
LCDCF\_WINON, [203](#)  
LY\_REG, [215](#)  
LYC\_REG, [215](#)  
MEMCTL\_BASEOFF, [232](#)  
MEMCTL\_BASEON, [232](#)  
MEMCTL\_CROMOFF, [232](#)  
MEMCTL\_CROMON, [232](#)  
MEMCTL\_EXTOFF, [232](#)  
MEMCTL\_EXTON, [232](#)  
MEMCTL\_JOYOFF, [232](#)  
MEMCTL\_JOYON, [232](#)  
MEMCTL\_RAMOFF, [232](#)  
MEMCTL\_RAMON, [232](#)  
MEMCTL\_ROMOFF, [232](#)  
MEMCTL\_ROMON, [232](#)  
NR10\_REG, [213](#)  
NR11\_REG, [213](#)  
NR12\_REG, [213](#)

NR13\_REG, [213](#)  
NR14\_REG, [213](#)  
NR21\_REG, [213](#)  
NR22\_REG, [213](#)  
NR23\_REG, [213](#)  
NR24\_REG, [213](#)  
NR30\_REG, [214](#)  
NR31\_REG, [214](#)  
NR32\_REG, [214](#)  
NR33\_REG, [214](#)  
NR34\_REG, [214](#)  
NR41\_REG, [214](#)  
NR42\_REG, [214](#)  
NR43\_REG, [214](#)  
NR44\_REG, [214](#)  
NR50\_REG, [214](#)  
NR51\_REG, [214](#)  
NR52\_REG, [214](#)  
OAMF\_BANK0, [210](#)  
OAMF\_BANK1, [210](#)  
OAMF\_CGB\_PAL0, [210](#)  
OAMF\_CGB\_PAL1, [210](#)  
OAMF\_CGB\_PAL2, [210](#)  
OAMF\_CGB\_PAL3, [210](#)  
OAMF\_CGB\_PAL4, [210](#)  
OAMF\_CGB\_PAL5, [210](#)  
OAMF\_CGB\_PAL6, [210](#)  
OAMF\_CGB\_PAL7, [211](#)  
OAMF\_PAL0, [210](#)  
OAMF\_PAL1, [210](#)  
OAMF\_PALMASK, [211](#)  
OAMF\_PRI, [210](#)  
OAMF\_XFLIP, [210](#)  
OAMF\_YFLIP, [210](#)  
OBP0\_REG, [215](#)  
OBP1\_REG, [215](#)  
OCPD\_REG, [216](#)  
OCPS\_REG, [216](#)  
OCPSF\_AUTOINC, [208](#)  
P1\_REG, [212](#)  
P1F\_0, [199](#)  
P1F\_1, [199](#)  
P1F\_2, [199](#)  
P1F\_3, [199](#)  
P1F\_4, [199](#)  
P1F\_5, [199](#)  
P1F\_GET\_BTN, [199](#)  
P1F\_GET\_DPAD, [199](#)  
P1F\_GET\_NONE, [199](#)  
PCM12\_REG, [216](#)  
PCM34\_REG, [216](#)  
PCM\_SAMPLE, [214](#)  
PPUCTRL\_BG\_CHR, [225](#)  
PPUCTRL\_INC32, [225](#)  
PPUCTRL\_NMI, [224](#)  
PPUCTRL\_SPR\_8X16, [224](#)  
PPUCTRL\_SPR\_8X8, [224](#)  
PPUCTRL\_SPR\_CHR, [225](#)  
PPUMASK\_BLUE, [225](#)  
PPUMASK\_GREEN, [225](#)  
PPUMASK\_MONOCHROME, [225](#)  
PPUMASK\_RED, [225](#)  
PPUMASK\_SHOW\_BG, [225](#)  
PPUMASK\_SHOW\_BG\_LC, [225](#)  
PPUMASK\_SHOW\_SPR, [225](#)  
PPUMASK\_SHOW\_SPR\_LC, [225](#)  
PSG\_CH0, [218](#), [233](#)  
PSG\_CH1, [218](#), [233](#)  
PSG\_CH2, [218](#), [234](#)  
PSG\_CH3, [218](#), [234](#)  
PSG\_LATCH, [218](#), [233](#)  
PSG\_VOLUME, [218](#), [234](#)  
R0\_CB\_INPUT, [219](#)  
R0\_CB\_OUTPUT, [219](#)  
R0\_DEFAULT, [219](#), [235](#)  
R0\_ES, [219](#), [235](#)  
R0\_ES\_OFF, [219](#), [235](#)  
R0\_HSCRL, [234](#)  
R0\_HSCRL\_INH, [234](#)  
R0\_IE1, [219](#), [234](#)  
R0\_IE1\_OFF, [219](#), [234](#)  
R0\_IE2, [219](#)  
R0\_IE2\_OFF, [219](#)  
R0\_LCB, [234](#)  
R0\_NO\_LCB, [234](#)  
R0\_SCR\_MODE1, [219](#)  
R0\_SCR\_MODE2, [219](#)  
R0\_SCR\_MODE3, [219](#)  
R0\_SS, [234](#)  
R0\_SS\_OFF, [234](#)  
R0\_VSCRL, [234](#)  
R0\_VSCRL\_INH, [234](#)  
R10\_INT\_EVERY, [222](#), [237](#)  
R10\_INT\_OFF, [222](#), [237](#)  
R1\_DEFAULT, [220](#), [235](#)  
R1\_DISP\_OFF, [220](#), [235](#)  
R1\_DISP\_ON, [220](#), [235](#)  
R1\_IE, [220](#), [235](#)  
R1\_IE\_OFF, [220](#), [235](#)  
R1\_SCR\_MODE1, [220](#)  
R1\_SCR\_MODE2, [220](#)  
R1\_SCR\_MODE3, [220](#)  
R1\_SPR\_16X16, [220](#)  
R1\_SPR\_8X16, [235](#)  
R1\_SPR\_8X8, [220](#), [235](#)  
R1\_SPR\_MAG, [220](#)  
R1\_SPR\_MAG\_OFF, [220](#)  
R2\_MAP\_0x0000, [221](#), [236](#)  
R2\_MAP\_0x0800, [221](#), [236](#)  
R2\_MAP\_0x1000, [221](#), [235](#)  
R2\_MAP\_0x1800, [220](#), [235](#)  
R2\_MAP\_0x2000, [220](#), [235](#)  
R2\_MAP\_0x2800, [220](#), [235](#)  
R2\_MAP\_0x3000, [220](#), [235](#)  
R2\_MAP\_0x3800, [220](#), [235](#)  
R5\_SAT\_0x1F00, [236](#)

[R5\\_SAT\\_0x3F00](#), [221](#), [236](#)  
[R5\\_SAT\\_MASK](#), [221](#), [236](#)  
[R6\\_BANK0](#), [221](#), [236](#)  
[R6\\_BANK1](#), [221](#), [236](#)  
[R6\\_DATA\\_0x0000](#), [221](#), [236](#)  
[R6\\_DATA\\_0x2000](#), [221](#), [236](#)  
[R7\\_COLOR\\_MASK](#), [221](#), [236](#)  
[RAMCTL\\_BANK](#), [238](#)  
[RAMCTL\\_PROT](#), [239](#)  
[RAMCTL\\_RAM](#), [239](#)  
[RAMCTL\\_RO](#), [239](#)  
[RAMCTL\\_ROM](#), [238](#)  
[rAUD1ENV](#), [201](#)  
[rAUD1HIGH](#), [201](#)  
[rAUD1LEN](#), [201](#)  
[rAUD1LOW](#), [201](#)  
[rAUD1SWEEP](#), [201](#)  
[rAUD2ENV](#), [201](#)  
[rAUD2HIGH](#), [201](#)  
[rAUD2LEN](#), [201](#)  
[rAUD2LOW](#), [201](#)  
[rAUD3ENA](#), [201](#)  
[rAUD3HIGH](#), [202](#)  
[rAUD3LEN](#), [202](#)  
[rAUD3LEVEL](#), [202](#)  
[rAUD3LOW](#), [202](#)  
[rAUD4ENV](#), [202](#)  
[rAUD4GO](#), [202](#)  
[rAUD4LEN](#), [202](#)  
[rAUD4POLY](#), [202](#)  
[rAUDENA](#), [203](#)  
[rAUDTERM](#), [202](#)  
[rAUDVOL](#), [202](#)  
[rBCPD](#), [208](#)  
[rBCPS](#), [208](#)  
[rBGP](#), [206](#)  
[rDIV](#), [200](#)  
[rDMA](#), [206](#)  
[rHDMA1](#), [207](#)  
[rHDMA2](#), [207](#)  
[rHDMA3](#), [208](#)  
[rHDMA4](#), [208](#)  
[rHDMA5](#), [208](#)  
[rIE](#), [209](#)  
[rIF](#), [201](#)  
[rKEY1](#), [206](#)  
[rLDCD](#), [203](#)  
[rLY](#), [206](#)  
[rLYC](#), [206](#)  
[rOBP0](#), [206](#)  
[rOBP1](#), [206](#)  
[rOCPD](#), [208](#)  
[rOCPs](#), [208](#)  
[rP1](#), [199](#)  
[RP\\_REG](#), [216](#)  
[rPCM12](#), [209](#)  
[rPCM34](#), [209](#)  
[RPF\\_DATAIN](#), [208](#)  
[RPF\\_ENREAD](#), [208](#)  
[RPF\\_WRITE\\_HI](#), [208](#)  
[RPF\\_WRITE\\_LO](#), [208](#)  
[rRAMB](#), [212](#)  
[rRAMG](#), [212](#)  
[rROMB0](#), [212](#)  
[rROMB1](#), [212](#)  
[rRP](#), [208](#)  
[rSB](#), [199](#)  
[rSC](#), [199](#)  
[rSCX](#), [206](#)  
[rSCY](#), [206](#)  
[rSMBK](#), [209](#)  
[rSPD](#), [206](#)  
[rSTAT](#), [205](#)  
[rSVBK](#), [208](#)  
[rTAC](#), [200](#)  
[rTIMA](#), [200](#)  
[rTMA](#), [200](#)  
[rVBK](#), [206](#)  
[rWX](#), [206](#)  
[rWY](#), [206](#)  
[SB\\_REG](#), [213](#)  
[SC\\_REG](#), [213](#)  
[SCF\\_SOURCE](#), [200](#)  
[SCF\\_SPEED](#), [200](#)  
[SCF\\_START](#), [200](#)  
[SCX\\_REG](#), [214](#)  
[SCY\\_REG](#), [214](#)  
[shadow\\_PPUCTRL](#), [227](#)  
[shadow\\_PPUMASK](#), [227](#)  
[shadow\\_VDP\\_R0](#), [222](#), [239](#)  
[shadow\\_VDP\\_R1](#), [222](#), [239](#)  
[shadow\\_VDP\\_R10](#), [223](#), [240](#)  
[shadow\\_VDP\\_R2](#), [222](#), [239](#)  
[shadow\\_VDP\\_R3](#), [222](#), [239](#)  
[shadow\\_VDP\\_R4](#), [222](#), [239](#)  
[shadow\\_VDP\\_R5](#), [223](#), [239](#)  
[shadow\\_VDP\\_R6](#), [223](#), [239](#)  
[shadow\\_VDP\\_R7](#), [223](#), [239](#)  
[shadow\\_VDP\\_R8](#), [223](#), [240](#)  
[shadow\\_VDP\\_R9](#), [223](#), [240](#)  
[shadow\\_VDP\\_RBORDER](#), [223](#), [239](#)  
[shadow\\_VDP\\_RSCX](#), [223](#), [240](#)  
[shadow\\_VDP\\_RSCY](#), [223](#), [240](#)  
[SIOCTL\\_BS0](#), [231](#)  
[SIOCTL\\_BS1](#), [231](#)  
[SIOCTL\\_FRER](#), [231](#)  
[SIOCTL\\_INT](#), [231](#)  
[SIOCTL\\_RON](#), [231](#)  
[SIOCTL\\_RXRD](#), [231](#)  
[SIOCTL\\_TON](#), [231](#)  
[SIOCTL\\_TXFL](#), [231](#)  
[SIOF\\_B\\_CLOCK](#), [200](#)  
[SIOF\\_B\\_SPEED](#), [200](#)  
[SIOF\\_B\\_XFER\\_START](#), [200](#)  
[SIOF\\_CLOCK\\_EXT](#), [200](#)  
[SIOF\\_CLOCK\\_INT](#), [200](#)



SIOF\_SPEED\_1X, [200](#)  
SIOF\_SPEED\_32X, [200](#)  
SIOF\_XFER\_START, [199](#)  
SOUNDPAN\_NOSL, [231](#)  
SOUNDPAN\_NOSR, [231](#)  
SOUNDPAN\_TN1L, [231](#)  
SOUNDPAN\_TN1R, [231](#)  
SOUNDPAN\_TN2L, [231](#)  
SOUNDPAN\_TN2R, [231](#)  
SOUNDPAN\_TN3L, [231](#)  
SOUNDPAN\_TN3R, [231](#)  
STAT\_REG, [214](#)  
STATF\_9\_SPR, [219](#), [234](#)  
STATF\_B\_BUSY, [206](#)  
STATF\_B\_LYC, [205](#)  
STATF\_B\_LYCF, [205](#)  
STATF\_B\_MODE00, [205](#)  
STATF\_B\_MODE01, [205](#)  
STATF\_B\_MODE10, [205](#)  
STATF\_B\_OAM, [206](#)  
STATF\_B\_VBL, [205](#)  
STATF\_BUSY, [205](#)  
STATF\_HBL, [205](#)  
STATF\_INT\_VBL, [219](#), [234](#)  
STATF\_LCD, [205](#)  
STATF\_LYC, [205](#)  
STATF\_LYCF, [205](#)  
STATF\_MODE00, [205](#)  
STATF\_MODE01, [205](#)  
STATF\_MODE10, [205](#)  
STATF\_OAM, [205](#)  
STATF\_SPR\_COLL, [219](#), [234](#)  
STATF\_VBL, [205](#)  
SVBK\_REG, [216](#)  
SYSTEM\_NTSC, [222](#)  
SYSTEM\_PAL, [222](#)  
TAC\_REG, [213](#)  
TACF\_16KHZ, [200](#)  
TACF\_262KHZ, [201](#)  
TACF\_4KHZ, [200](#)  
TACF\_65KHZ, [201](#)  
TACF\_START, [200](#)  
TACF\_STOP, [200](#)  
TIMA\_REG, [213](#)  
TMA\_REG, [213](#)  
VBK\_ATTRIBUTES, [207](#), [222](#), [239](#)  
VBK\_BANK\_0, [206](#)  
VBK\_BANK\_1, [207](#)  
VBK\_REG, [215](#)  
VBK\_TILES, [207](#), [222](#), [239](#)  
VDP\_ATTR\_SHIFT, [223](#), [240](#)  
VDP\_R0, [219](#), [234](#)  
VDP\_R1, [219](#), [235](#)  
VDP\_R10, [222](#), [237](#)  
VDP\_R2, [220](#), [235](#)  
VDP\_R3, [221](#), [236](#)  
VDP\_R4, [221](#), [236](#)  
VDP\_R5, [221](#), [236](#)  
VDP\_R6, [221](#), [236](#)  
VDP\_R7, [221](#), [236](#)  
VDP\_R8, [221](#), [236](#)  
VDP\_R9, [222](#), [237](#)  
VDP\_RBORDER, [221](#), [236](#)  
VDP\_REG\_MASK, [219](#), [234](#)  
VDP\_RSCX, [221](#), [236](#)  
VDP\_RSCY, [222](#), [237](#)  
VDP\_SAT\_TERM, [222](#), [239](#)  
WX\_REG, [215](#)  
WY\_REG, [215](#)  
HARDWARE\_SPRITE\_CAN\_FLIP\_X  
    gb.h, [154](#)  
    msx.h, [287](#)  
    nes.h, [314](#)  
    sms.h, [355](#)  
HARDWARE\_SPRITE\_CAN\_FLIP\_Y  
    gb.h, [155](#)  
    msx.h, [287](#)  
    nes.h, [314](#)  
    sms.h, [355](#)  
hblank\_copy  
    hblankcpy.h, [241](#)  
hblank\_copy\_destination  
    hblankcpy.h, [241](#)  
hblank\_copy\_vram  
    hblankcpy.h, [240](#)  
hblank\_cpy\_vram  
    hblankcpy.h, [241](#)  
hblankcpy.h  
    hblank\_copy, [241](#)  
    hblank\_copy\_destination, [241](#)  
    hblank\_copy\_vram, [240](#)  
    hblank\_cpy\_vram, [241](#)  
HDMA1\_REG  
    hardware.h, [215](#)  
HDMA2\_REG  
    hardware.h, [215](#)  
HDMA3\_REG  
    hardware.h, [215](#)  
HDMA4\_REG  
    hardware.h, [216](#)  
HDMA5\_REG  
    hardware.h, [216](#)  
HDMA5F\_BUSY  
    hardware.h, [208](#)  
HDMA5F\_MODE\_GP  
    hardware.h, [208](#)  
HDMA5F\_MODE\_HBL  
    hardware.h, [208](#)  
HIDE\_BKG  
    gb.h, [153](#)  
    msx.h, [284](#)  
    nes.h, [313](#)  
    sms.h, [352](#)  
HIDE\_LEFT\_COLUMN  
    gb.h, [153](#)  
    msx.h, [284](#)

- nes.h, [312](#)
- sms.h, [352](#)
- hide metasprite
  - metasprites.h, [249](#), [252](#), [257](#), [262](#)
- hide\_sprite
  - gb.h, [183](#)
  - msx.h, [300](#)
  - nes.h, [335](#)
  - sms.h, [368](#)
- HIDE\_SPRITES
  - gb.h, [153](#)
  - msx.h, [284](#)
  - nes.h, [313](#)
  - sms.h, [352](#)
- hide\_sprites\_range
  - metasprites.h, [246](#), [251](#), [254](#), [260](#)
- HIDE\_WIN
  - gb.h, [153](#)
  - msx.h, [284](#)
  - sms.h, [352](#)
- hiramcpy
  - gb.h, [162](#)
- IE\_REG
  - hardware.h, [216](#)
- IEF\_HILO
  - hardware.h, [209](#)
- IEF\_SERIAL
  - hardware.h, [209](#)
- IEF\_STAT
  - hardware.h, [209](#)
- IEF\_TIMER
  - hardware.h, [209](#)
- IEF\_VBLANK
  - hardware.h, [209](#)
- IF\_REG
  - hardware.h, [213](#)
- INCBIN
  - incbin.h, [273](#)
- incbin.h
  - BANK, [273](#)
  - INCBIN, [273](#)
  - INCBIN\_EXTERN, [272](#)
  - INCBIN\_SIZE, [273](#)
- INCBIN\_EXTERN
  - incbin.h, [272](#)
- INCBIN\_SIZE
  - incbin.h, [273](#)
- init\_bkg
  - gb.h, [187](#)
  - nes.h, [339](#)
- init\_win
  - gb.h, [187](#)
- initarand
  - rand.h, [343](#)
- initrand
  - rand.h, [342](#)
- INT16
  - types.h, [110](#), [111](#), [114](#)
- INT16\_C
  - stdint.h, [376](#)
- INT16\_MAX
  - stdint.h, [375](#)
- INT16\_MIN
  - stdint.h, [374](#)
- int16\_t
  - stdint.h, [377](#)
- INT32
  - types.h, [110](#), [111](#), [115](#)
- INT32\_C
  - stdint.h, [377](#)
- INT32\_MAX
  - stdint.h, [375](#)
- INT32\_MIN
  - stdint.h, [374](#)
- int32\_t
  - stdint.h, [377](#)
- INT8
  - types.h, [110](#), [111](#), [114](#)
- INT8\_C
  - stdint.h, [376](#)
- INT8\_MAX
  - stdint.h, [374](#)
- INT8\_MIN
  - stdint.h, [374](#)
- int8\_t
  - stdint.h, [377](#)
- INT\_FAST16\_MAX
  - stdint.h, [376](#)
- INT\_FAST16\_MIN
  - stdint.h, [375](#)
- int\_fast16\_t
  - stdint.h, [378](#)
- INT\_FAST32\_MAX
  - stdint.h, [376](#)
- INT\_FAST32\_MIN
  - stdint.h, [375](#)
- int\_fast32\_t
  - stdint.h, [378](#)
- INT\_FAST8\_MAX
  - stdint.h, [375](#)
- INT\_FAST8\_MIN
  - stdint.h, [375](#)
- int\_fast8\_t
  - stdint.h, [378](#)
- int\_handler
  - gb.h, [155](#)
  - msx.h, [287](#)
  - nes.h, [314](#)
  - sms.h, [356](#)
- INT\_LEAST16\_MAX
  - stdint.h, [375](#)
- INT\_LEAST16\_MIN
  - stdint.h, [375](#)
- int\_least16\_t
  - stdint.h, [378](#)
- INT\_LEAST32\_MAX

- stdint.h, [375](#)
- INT\_LEAST32\_MIN
  - stdint.h, [375](#)
- int\_least32\_t
  - stdint.h, [378](#)
- INT\_LEAST8\_MAX
  - stdint.h, [375](#)
- INT\_LEAST8\_MIN
  - stdint.h, [375](#)
- int\_least8\_t
  - stdint.h, [378](#)
- INT\_MAX
  - limits.h, [276](#)
- INT\_MIN
  - limits.h, [276](#)
- INTERRUPT
  - types.h, [113](#)
- INTMAX\_C
  - stdint.h, [377](#)
- INTMAX\_MAX
  - stdint.h, [376](#)
- INTMAX\_MIN
  - stdint.h, [376](#)
- intmax\_t
  - stdint.h, [378](#)
- INTPTR\_MAX
  - stdint.h, [376](#)
- INTPTR\_MIN
  - stdint.h, [376](#)
- intptr\_t
  - stdint.h, [378](#)
- IO\_ERROR
  - gb.h, [149](#)
- IO\_IDLE
  - gb.h, [149](#)
- IO\_RECEIVING
  - gb.h, [149](#)
- IO\_SENDING
  - gb.h, [149](#)
- isalpha
  - ctype.h, [116](#)
- isdigit
  - ctype.h, [117](#)
- islower
  - ctype.h, [117](#)
- isr.h
  - ISR\_NESTED\_VECTOR, [243](#)
  - isr\_nested\_vector\_t, [243](#)
  - ISR\_VECTOR, [242](#)
  - isr\_vector\_t, [243](#)
  - VECTOR\_JOYPAD, [242](#)
  - VECTOR\_SERIAL, [242](#)
  - VECTOR\_STAT, [242](#)
  - VECTOR\_TIMER, [242](#)
- ISR\_NESTED\_VECTOR
  - isr.h, [243](#)
- isr\_nested\_vector\_t, [90](#)
  - func, [90](#)
- isr.h, [243](#)
  - opcode, [90](#)
- ISR\_VECTOR
  - isr.h, [242](#)
- isr\_vector\_t, [90](#)
  - func, [91](#)
  - isr.h, [243](#)
  - opcode, [91](#)
- isspace
  - ctype.h, [117](#)
- isupper
  - ctype.h, [117](#)
- itoa
  - stdlib.h, [382](#)
- iyh
  - msx.h, [301](#)
  - sms.h, [370](#)
- iyi
  - metasprites.h, [252](#), [263](#)
  - msx.h, [301](#)
  - sms.h, [370](#)
- J\_A
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_B
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_DOWN
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_LEFT
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_RIGHT
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_SELECT
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_START
  - gb.h, [144](#)
  - msx.h, [281](#)
  - nes.h, [308](#)
  - sms.h, [349](#)
- J\_UP
  - gb.h, [144](#)

- msx.h, [281](#)
- nes.h, [308](#)
- sms.h, [348](#)
- jmp\_buf
  - setjmp.h, [344](#)
- joy0
  - joypads\_t, [92](#)
- joy1
  - joypads\_t, [92](#)
- joy2
  - joypads\_t, [92](#)
- joy3
  - joypads\_t, [92](#)
- JOY\_IFLAG
  - gb.h, [146](#)
  - msx.h, [283](#)
  - sms.h, [351](#)
- JOY\_P1\_DOWN
  - hardware.h, [237](#)
- JOY\_P1\_LEFT
  - hardware.h, [237](#)
- JOY\_P1\_LIGHT
  - hardware.h, [238](#)
- JOY\_P1\_MD\_A
  - hardware.h, [237](#)
- JOY\_P1\_MD\_MODE
  - hardware.h, [237](#)
- JOY\_P1\_MD\_START
  - hardware.h, [237](#)
- JOY\_P1\_MD\_X
  - hardware.h, [237](#)
- JOY\_P1\_MD\_Y
  - hardware.h, [237](#)
- JOY\_P1\_MD\_Z
  - hardware.h, [237](#)
- JOY\_P1\_RIGHT
  - hardware.h, [237](#)
- JOY\_P1\_SW1
  - hardware.h, [237](#)
- JOY\_P1\_SW2
  - hardware.h, [237](#)
- JOY\_P1\_TH\_DIR\_IN
  - hardware.h, [232](#)
- JOY\_P1\_TH\_DIR\_OUT
  - hardware.h, [232](#)
- JOY\_P1\_TH\_OUT\_HI
  - hardware.h, [233](#)
- JOY\_P1\_TH\_OUT\_LO
  - hardware.h, [233](#)
- JOY\_P1\_TR\_DIR\_IN
  - hardware.h, [232](#)
- JOY\_P1\_TR\_DIR\_OUT
  - hardware.h, [232](#)
- JOY\_P1\_TR\_OUT\_HI
  - hardware.h, [233](#)
- JOY\_P1\_TR\_OUT\_LO
  - hardware.h, [233](#)
- JOY\_P1\_TRIGGER
  - hardware.h, [237](#)
- JOY\_P2\_UP
  - hardware.h, [237](#)
- JOY\_P2\_DOWN
  - hardware.h, [238](#)
- JOY\_P2\_LEFT
  - hardware.h, [238](#)
- JOY\_P2\_LIGHT
  - hardware.h, [238](#)
- JOY\_P2\_MD\_A
  - hardware.h, [238](#)
- JOY\_P2\_MD\_MODE
  - hardware.h, [238](#)
- JOY\_P2\_MD\_START
  - hardware.h, [238](#)
- JOY\_P2\_MD\_X
  - hardware.h, [238](#)
- JOY\_P2\_MD\_Y
  - hardware.h, [238](#)
- JOY\_P2\_MD\_Z
  - hardware.h, [238](#)
- JOY\_P2\_RIGHT
  - hardware.h, [238](#)
- JOY\_P2\_SW1
  - hardware.h, [238](#)
- JOY\_P2\_SW2
  - hardware.h, [238](#)
- JOY\_P2\_TH\_DIR\_IN
  - hardware.h, [233](#)
- JOY\_P2\_TH\_DIR\_OUT
  - hardware.h, [233](#)
- JOY\_P2\_TH\_OUT\_HI
  - hardware.h, [233](#)
- JOY\_P2\_TH\_OUT\_LO
  - hardware.h, [233](#)
- JOY\_P2\_TR\_DIR\_IN
  - hardware.h, [232](#)
- JOY\_P2\_TR\_DIR\_OUT
  - hardware.h, [233](#)
- JOY\_P2\_TR\_OUT\_HI
  - hardware.h, [233](#)
- JOY\_P2\_TR\_OUT\_LO
  - hardware.h, [233](#)
- JOY\_P2\_TRIGGER
  - hardware.h, [238](#)
- JOY\_P2\_UP
  - hardware.h, [238](#)
- JOY\_RESET
  - hardware.h, [238](#)
- JOY\_TH\_HI
  - hardware.h, [233](#)
- JOY\_TH\_LO
  - hardware.h, [233](#)
- joypad
  - gb.h, [159](#)
  - msx.h, [291](#)
  - nes.h, [317](#)
  - sms.h, [358](#)

- joypad\_ex
  - gb.h, [161](#)
  - msx.h, [291](#)
  - nes.h, [318](#)
  - sms.h, [359](#)
- joypad\_init
  - gb.h, [160](#)
  - msx.h, [291](#)
  - nes.h, [317](#)
  - sms.h, [359](#)
- joypads
  - joypads\_t, [92](#)
- joypads\_t, [91](#)
  - joy0, [92](#)
  - joy1, [92](#)
  - joy2, [92](#)
  - joy3, [92](#)
  - joypads, [92](#)
  - npads, [92](#)
- KEY1\_REG
  - hardware.h, [215](#)
- KEY1F\_DBLSPED
  - hardware.h, [206](#)
- KEY1F\_PREPARE
  - hardware.h, [206](#)
- I
  - \_fixed, [89](#)
  - gb.h, [189](#)
  - msx.h, [301](#)
  - sms.h, [370](#)
- labs
  - stdlib.h, [381](#)
- LCD\_IFLAG
  - gb.h, [146](#)
  - msx.h, [282](#)
  - sms.h, [350](#)
- LCDC\_REG
  - hardware.h, [214](#)
- LCDCF\_B\_BG8000
  - hardware.h, [204](#)
- LCDCF\_B\_BG9C00
  - hardware.h, [204](#)
- LCDCF\_B\_BGON
  - hardware.h, [204](#)
- LCDCF\_B\_OBJ16
  - hardware.h, [204](#)
- LCDCF\_B\_OBJON
  - hardware.h, [204](#)
- LCDCF\_B\_ON
  - hardware.h, [204](#)
- LCDCF\_B\_WIN9C00
  - hardware.h, [204](#)
- LCDCF\_B\_WINON
  - hardware.h, [204](#)
- LCDCF\_BG8000
  - hardware.h, [204](#)
- LCDCF\_BG8800
  - hardware.h, [204](#)
- LCDCF\_BG9800
  - hardware.h, [204](#)
- LCDCF\_BG9C00
  - hardware.h, [204](#)
- LCDCF\_BGOFF
  - hardware.h, [204](#)
- LCDCF\_BGON
  - hardware.h, [204](#)
- LCDCF\_OBJ16
  - hardware.h, [204](#)
- LCDCF\_OBJ8
  - hardware.h, [204](#)
- LCDCF\_OBJOFF
  - hardware.h, [204](#)
- LCDCF\_OBJON
  - hardware.h, [204](#)
- LCDCF\_OFF
  - hardware.h, [203](#)
- LCDCF\_ON
  - hardware.h, [203](#)
- LCDCF\_WIN9800
  - hardware.h, [203](#)
- LCDCF\_WIN9C00
  - hardware.h, [203](#)
- LCDCF\_WINOFF
  - hardware.h, [203](#)
- LCDCF\_WINON
  - hardware.h, [203](#)
- limits.h
  - CHAR\_BIT, [275](#)
  - CHAR\_MAX, [276](#)
  - CHAR\_MIN, [276](#)
  - INT\_MAX, [276](#)
  - INT\_MIN, [276](#)
  - LONG\_MAX, [276](#)
  - LONG\_MIN, [276](#)
  - SCHAR\_MAX, [275](#)
  - SCHAR\_MIN, [276](#)
  - SHRT\_MAX, [276](#)
  - SHRT\_MIN, [276](#)
  - UCHAR\_MAX, [276](#)
  - UINT\_MAX, [276](#)
  - UINT\_MIN, [276](#)
  - ULONG\_MAX, [276](#)
  - ULONG\_MIN, [276](#)
  - USHRT\_MAX, [276](#)
  - USHRT\_MIN, [276](#)
- line
  - drawing.h, [134](#)
- List of gbdk fonts, [88](#)
  - font\_ibm, [88](#)
  - font\_ibm\_fixed, [88](#)
  - font\_italic, [88](#)
  - font\_min, [88](#)
  - font\_spect, [88](#)
- LONG\_MAX
  - limits.h, [276](#)

- LONG\_MIN
  - limits.h, 276
- longjmp
  - setjmp.h, 344
- LTGREY
  - drawing.h, 132
- ltoa
  - stdlib.h, 382
- LWORD
  - types.h, 113
- LY\_REG
  - hardware.h, 215
- LYC\_REG
  - hardware.h, 215
- M\_DRAWING
  - gb.h, 144
  - nes.h, 308
- M\_FILL
  - drawing.h, 132
- M\_NO\_INTERP
  - gb.h, 145
  - msx.h, 281
  - nes.h, 309
  - sms.h, 349
- M\_NO\_SCROLL
  - gb.h, 144
  - msx.h, 281
  - nes.h, 309
  - sms.h, 349
- M\_NOFILL
  - drawing.h, 132
- M\_TEXT\_INOUT
  - gb.h, 144
  - msx.h, 281
  - nes.h, 308
  - sms.h, 349
- M\_TEXT\_OUT
  - gb.h, 144
  - msx.h, 281
  - nes.h, 308
  - sms.h, 349
- MAKE\_BCD
  - bcd.h, 118, 121, 123
- malloc
  - stdlib.h, 383
- MAX\_HARDWARE\_SPRITES
  - gb.h, 154
  - msx.h, 287
  - nes.h, 314
  - sms.h, 355
- MAX\_LCD\_ISR\_CALLS
  - nes.h, 310
- MAXWNDPOSX
  - gb.h, 147
  - msx.h, 283
  - sms.h, 351
- MAXWNDPOSY
  - gb.h, 148
- msx.h, 283
- sms.h, 351
- memcmp
  - string.h, 102, 105, 109
- memcpy
  - string.h, 99, 103, 107
- MEMCTL\_BASEOFF
  - hardware.h, 232
- MEMCTL\_BASEON
  - hardware.h, 232
- MEMCTL\_CROMOFF
  - hardware.h, 232
- MEMCTL\_CROMON
  - hardware.h, 232
- MEMCTL\_EXTOFF
  - hardware.h, 232
- MEMCTL\_EXTON
  - hardware.h, 232
- MEMCTL\_JOYOFF
  - hardware.h, 232
- MEMCTL\_JOYON
  - hardware.h, 232
- MEMCTL\_RAMOFF
  - hardware.h, 232
- MEMCTL\_RAMON
  - hardware.h, 232
- MEMCTL\_ROMOFF
  - hardware.h, 232
- MEMCTL\_ROMON
  - hardware.h, 232
- memmove
  - string.h, 100, 103, 107
- memset
  - string.h, 100, 104, 107
- METASPR\_ITEM
  - metasprites.h, 245, 250, 254, 259
- METASPR\_TERM
  - metasprites.h, 245, 251, 254, 259
- metasprite\_end
  - metasprites.h, 245, 250, 254, 259
- metasprite\_t, 92
  - dtile, 93
  - dx, 93
  - dy, 93
  - metasprites.h, 245, 251, 254, 259
  - props, 93
- metasprites.h
  - \_\_current\_base\_prop, 249, 258
  - \_\_current\_base\_tile, 249, 252, 258, 263
  - \_\_current\_metasprite, 249, 252, 258, 263
  - \_\_render\_shadow\_OAM, 250, 252, 258, 263
  - hide\_metasprite, 249, 252, 257, 262
  - hide\_sprites\_range, 246, 251, 254, 260
  - iyI, 252, 263
  - METASPR\_ITEM, 245, 250, 254, 259
  - METASPR\_TERM, 245, 251, 254, 259
  - metasprite\_end, 245, 250, 254, 259
  - metasprite\_t, 245, 251, 254, 259

- move metasprite, [247](#), [252](#), [255](#), [260](#)
  - move metasprite\_ex, [246](#), [251](#), [254](#), [260](#)
  - move metasprite\_flipx, [247](#), [255](#), [260](#)
  - move metasprite\_flipy, [248](#), [257](#), [262](#)
  - move metasprite\_hflip, [248](#), [256](#)
  - move metasprite\_hvflip, [249](#), [257](#)
  - move metasprite\_vflip, [247](#), [256](#)
- mfont\_handle
  - font.h, [271](#)
- MGB\_TYPE
  - gb.h, [148](#)
- MINWNDPOSX
  - gb.h, [147](#)
  - msx.h, [283](#)
  - sms.h, [351](#)
- MINWNPYSY
  - gb.h, [147](#)
  - msx.h, [283](#)
  - sms.h, [351](#)
- mode
  - gb.h, [158](#)
  - msx.h, [288](#)
  - nes.h, [316](#)
  - sms.h, [356](#)
- move\_bkg
  - gb.h, [173](#)
  - msx.h, [290](#)
  - nes.h, [329](#)
  - sms.h, [358](#)
- move\_metasprite
  - metasprites.h, [247](#), [252](#), [255](#), [260](#)
- move\_metasprite\_ex
  - metasprites.h, [246](#), [251](#), [254](#), [260](#)
- move\_metasprite\_flipx
  - metasprites.h, [247](#), [255](#), [260](#)
- move\_metasprite\_flipy
  - metasprites.h, [248](#), [257](#), [262](#)
- move\_metasprite\_hflip
  - metasprites.h, [247](#), [256](#), [261](#)
- move\_metasprite\_hvflip
  - metasprites.h, [248](#), [256](#)
- move\_metasprite\_hvflip
  - metasprites.h, [249](#), [257](#)
- move\_metasprite\_vflip
  - metasprites.h, [247](#), [256](#)
- move\_sprite
  - gb.h, [182](#)
  - msx.h, [299](#)
  - nes.h, [334](#)
  - sms.h, [367](#)
- move\_win
  - gb.h, [179](#)
- MSX
  - msx.h, [280](#)
- msx.h
  - \_SYSTEM, [301](#)
  - READ VDP REG, [282](#)
  - \_WRITE\_VDP\_REG\_UNSAFE, [282](#)
  - \_current\_1bpp\_colors, [302](#)
  - \_current\_2bpp\_palette, [302](#)
  - \_current\_bank, [301](#)
  - \_map\_tile\_offset, [302](#)
  - \_shadow\_OAM\_OFF, [302](#)
  - \_shadow\_OAM\_base, [302](#)
  - \_submap\_tile\_offset, [302](#)
  - add\_JOY, [289](#)
  - add\_LCD, [289](#)
  - add\_SIO, [289](#)
  - add\_TIM, [289](#)
  - add\_VBL, [289](#)
  - b, [302](#)
  - BANK, [285](#)
  - BANKREF, [285](#)
  - BANKREF\_EXTERN, [285](#)
  - c, [301](#)
  - cancel\_pending\_interrupts, [290](#)
  - COMPAT\_PALETTE, [287](#)
  - cpu\_fast, [292](#)
  - CURRENT\_BANK, [285](#)
  - d, [301](#)
  - delay, [291](#)
  - DEVICE\_SUPPORTS\_COLOR, [285](#)
  - disable\_interrupts, [292](#)
  - DISABLE\_RAM, [286](#)
  - DISABLE\_VBL\_TRANSFER, [287](#)
  - DISPLAY\_OFF, [284](#)
  - display\_off, [290](#)
  - DISPLAY\_ON, [283](#)
  - DIV\_REG, [285](#)
  - e, [301](#)
  - EMPTY\_IFLAG, [282](#)
  - enable\_interrupts, [291](#)
  - ENABLE\_RAM, [286](#)
  - ENABLE\_VBL\_TRANSFER, [287](#)
  - fill\_bkg\_rect, [287](#)
  - fill\_rect, [296](#)
  - fill\_win\_rect, [287](#)
  - get\_bkg\_xy\_addr, [301](#)
  - get\_mode, [288](#)
  - get\_r\_reg, [290](#)
  - get\_sprite\_prop, [299](#)
  - get\_sprite\_tile, [297](#)
  - get\_system, [288](#)
  - get\_win\_xy\_addr, [287](#)
  - h, [301](#)
  - HARDWARE\_SPRITE\_CAN\_FLIP\_X, [287](#)
  - HARDWARE\_SPRITE\_CAN\_FLIP\_Y, [287](#)
  - HIDE\_BKG, [284](#)
  - HIDE\_LEFT\_COLUMN, [284](#)
  - hide\_sprite, [300](#)
  - HIDE\_SPRITES, [284](#)
  - HIDE\_WIN, [284](#)
  - int\_handler, [287](#)
  - iuh, [301](#)

- iyi, 301
- J\_A, 281
- J\_B, 281
- J\_DOWN, 281
- J\_LEFT, 281
- J\_RIGHT, 281
- J\_SELECT, 281
- J\_START, 281
- J\_UP, 281
- JOY\_IFLAG, 283
- joypad, 291
- joypad\_ex, 291
- joypad\_init, 291
- l, 301
- LCD\_IFLAG, 282
- M\_NO\_INTERP, 281
- M\_NO\_SCROLL, 281
- M\_TEXT\_INOUT, 281
- M\_TEXT\_OUT, 281
- MAX\_HARDWARE\_SPRITES, 287
- MAXWNDPOSX, 283
- MAXWNDPOSY, 283
- MINWNDPOSX, 283
- MINWNDPOSY, 283
- mode, 288
- move\_bkg, 290
- move\_sprite, 299
- MSX, 280
- OAM\_item\_t, 287
- refresh\_OAM, 290
- remove\_JOY, 289
- remove\_LCD, 289
- remove\_SIO, 289
- remove\_TIM, 289
- remove\_VBL, 289
- S\_BANK, 282
- S\_FLIPX, 282
- S\_FLIPY, 282
- S\_PAL, 282
- S\_PALETTE, 282
- S\_PRIORITY, 282
- SCREENHEIGHT, 283
- SCREENWIDTH, 283
- scroll\_bkg, 290
- scroll\_sprite, 299
- set\_1bpp\_colors, 293
- set\_2bpp\_palette, 293
- set\_attributed\_tile\_xy, 300
- set\_bkg\_1bpp\_data, 293
- set\_bkg\_4bpp\_data, 293
- set\_bkg\_based\_submap, 296
- set\_bkg\_based\_tiles, 294
- set\_bkg\_data, 293
- set\_bkg\_palette, 286
- set\_bkg\_palette\_entry, 286
- set\_bkg\_submap, 295
- set\_bkg\_tile\_xy, 287
- set\_bkg\_tiles, 287
- SET\_BORDER\_COLOR, 284
- set\_data, 294
- set\_default\_palette, 292
- set\_interrupts, 288
- set\_native\_sprite\_data, 293
- set\_native\_tile\_data, 292
- set\_palette, 292
- set\_palette\_entry, 292
- SET\_SHADOW\_OAM\_ADDRESS, 297
- set\_sprite\_1bpp\_data, 293
- set\_sprite\_data, 293
- set\_sprite\_palette, 286
- set\_sprite\_palette\_entry, 286
- set\_sprite\_prop, 297
- set\_sprite\_tile, 297
- set\_tile\_1bpp\_data, 293
- set\_tile\_map, 294
- set\_tile\_submap, 294
- set\_tile\_submap\_compat, 294
- set\_tile\_xy, 300
- set\_vram\_byte, 300
- set\_win\_based\_submap, 296
- set\_win\_based\_tiles, 294
- set\_win\_submap, 295
- set\_win\_tile\_xy, 287
- set\_win\_tiles, 287
- shadow\_OAM, 302
- SHOW\_BKG, 284
- SHOW\_LEFT\_COLUMN, 284
- SHOW\_SPRITES, 284
- SHOW\_WIN, 284
- SIO\_IFLAG, 283
- SPRITES\_16x16, 284
- SPRITES\_8x8, 285
- SWITCH\_RAM, 286
- SWITCH\_ROM, 290
- SWITCH\_ROM1, 286
- SWITCH\_ROM2, 286
- sys\_time, 301
- SYSTEM\_50HZ, 280
- SYSTEM\_60HZ, 280
- TIM\_IFLAG, 283
- VBK\_REG, 280
- VBL\_IFLAG, 282
- vmemcpy, 294
- vsync, 290
- wait\_vbl\_done, 290
- waitpad, 291
- waitpadup, 291
- WRITE\_VDP\_CMD, 288
- WRITE\_VDP\_DATA, 288
- NAKED
  - types.h, 113
- nes.h
  - \_SYSTEM, 340
  - \_current\_1bpp\_colors, 341
  - \_current\_bank, 340
  - \_map\_tile\_offset, 341



[\\_shadow\\_OAM\\_base](#), 341  
[\\_submap\\_tile\\_offset](#), 341  
[\\_switch\\_prg0](#), 340  
[add\\_LCD](#), 316  
[add\\_VBL](#), 315  
[BANK](#), 310  
[BANKREF](#), 311  
[BANKREF\\_EXTERN](#), 311  
[COMPAT\\_PALETTE](#), 313  
[CURRENT\\_BANK](#), 310  
[delay](#), 316  
[disable\\_interrupts](#), 319  
[DISABLE\\_OAM\\_DMA](#), 313  
[DISABLE\\_RAM](#), 312  
[DISABLE\\_VBL\\_TRANSFER](#), 314  
[DISPLAY\\_OFF](#), 312  
[display\\_off](#), 319  
[DISPLAY\\_ON](#), 312  
[display\\_on](#), 319  
[DMG\\_BLACK](#), 309  
[DMG\\_DARK\\_GRAY](#), 309  
[DMG\\_LITE\\_GRAY](#), 310  
[DMG\\_PALETTE](#), 310  
[DMG\\_WHITE](#), 310  
[enable\\_interrupts](#), 318  
[ENABLE\\_OAM\\_DMA](#), 314  
[ENABLE\\_RAM](#), 312  
[ENABLE\\_VBL\\_TRANSFER](#), 314  
[fill\\_bkg\\_rect](#), 340  
[fill\\_rect](#), 314  
[flush\\_shadow\\_attributes](#), 340  
[get\\_bkg\\_tile\\_xy](#), 329  
[get\\_bkg\\_tiles](#), 327  
[get\\_bkg\\_xy\\_addr](#), 320  
[get\\_mode](#), 316  
[get\\_sprite\\_prop](#), 334  
[get\\_sprite\\_tile](#), 332  
[get\\_system](#), 316  
[HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_X](#), 314  
[HARDWARE\\_SPRITE\\_CAN\\_FLIP\\_Y](#), 314  
[HIDE\\_BKG](#), 313  
[HIDE\\_LEFT\\_COLUMN](#), 312  
[hide\\_sprite](#), 335  
[HIDE\\_SPRITES](#), 313  
[init\\_bkg](#), 339  
[int\\_handler](#), 314  
[J\\_A](#), 308  
[J\\_B](#), 308  
[J\\_DOWN](#), 308  
[J\\_LEFT](#), 308  
[J\\_RIGHT](#), 308  
[J\\_SELECT](#), 308  
[J\\_START](#), 308  
[J\\_UP](#), 308  
[joypad](#), 317  
[joypad\\_ex](#), 318  
[joypad\\_init](#), 317  
[M\\_DRAWING](#), 308  
[M\\_NO\\_INTERP](#), 309  
[M\\_NO\\_SCROLL](#), 309  
[M\\_TEXT\\_INOUT](#), 308  
[M\\_TEXT\\_OUT](#), 308  
[MAX\\_HARDWARE\\_SPRITES](#), 314  
[MAX\\_LCD\\_ISR\\_CALLS](#), 310  
[mode](#), 316  
[move\\_bkg](#), 329  
[move\\_sprite](#), 334  
[NINTENDO\\_NES](#), 306  
[OAM\\_item\\_t](#), 314  
[palette\\_color\\_t](#), 314  
[refresh\\_OAM](#), 320  
[remove\\_LCD](#), 315  
[remove\\_VBL](#), 315  
[RGB](#), 307  
[RGB8](#), 307  
[RGB\\_AQUA](#), 307  
[RGB\\_BLACK](#), 308  
[RGB\\_BLUE](#), 307  
[RGB\\_CYAN](#), 307  
[RGB\\_DARKBLUE](#), 307  
[RGB\\_DARKGRAY](#), 308  
[RGB\\_DARKGREEN](#), 307  
[RGB\\_DARKRED](#), 307  
[RGB\\_DARKYELLOW](#), 307  
[RGB\\_GREEN](#), 307  
[RGB\\_LIGHTGRAY](#), 308  
[RGB\\_PINK](#), 307  
[RGB\\_PURPLE](#), 307  
[RGB\\_RED](#), 307  
[RGB\\_WHITE](#), 308  
[RGB\\_YELLOW](#), 307  
[RGBHTML](#), 307  
[S\\_FLIPX](#), 309  
[S\\_FLIPY](#), 309  
[S\\_PAL](#), 309  
[S\\_PALETTE](#), 309  
[S\\_PRIORITY](#), 309  
[SCREENHEIGHT](#), 310  
[SCREENWIDTH](#), 310  
[scroll\\_bkg](#), 330  
[scroll\\_sprite](#), 334  
[set\\_1bpp\\_colors](#), 320  
[set\\_1bpp\\_colors\\_ex](#), 320  
[set\\_2bpp\\_palette](#), 320  
[set\\_attribute\\_xy](#), 313  
[set\\_bkg\\_1bpp\\_data](#), 321  
[set\\_bkg\\_2bpp\\_data](#), 313  
[set\\_bkg\\_attribute\\_xy](#), 328  
[set\\_bkg\\_attribute\\_xy\\_nes16x16](#), 328  
[set\\_bkg\\_attributes](#), 324  
[set\\_bkg\\_attributes\\_nes16x16](#), 323  
[set\\_bkg\\_based\\_submap](#), 327  
[set\\_bkg\\_based\\_tiles](#), 325  
[set\\_bkg\\_data](#), 320  
[set\\_bkg\\_native\\_data](#), 336  
[set\\_bkg\\_palette](#), 314

- set\_bkg\_palette\_entry, 315
- set\_bkg\_submap, 326
- set\_bkg\_submap\_attributes, 325
- set\_bkg\_submap\_attributes\_nes16x16, 324
- set\_bkg\_tile\_xy, 328
- set\_bkg\_tiles, 322
- SET\_BORDER\_COLOR, 313
- set\_data, 336
- set\_native\_tile\_data, 339
- SET\_SHADOW\_OAM\_ADDRESS, 331
- set\_sprite\_1bpp\_data, 331
- set\_sprite\_2bpp\_data, 313
- set\_sprite\_data, 330
- set\_sprite\_native\_data, 338
- set\_sprite\_palette, 314
- set\_sprite\_palette\_entry, 315
- set\_sprite\_prop, 332
- set\_sprite\_tile, 331
- set\_tile\_data, 336
- set\_tile\_map, 313
- set\_tile\_submap, 313
- set\_tile\_xy, 313
- set\_tiles, 336
- set\_vram\_byte, 320
- shadow\_OAM, 341
- SHOW\_BKG, 313
- SHOW\_LEFT\_COLUMN, 312
- SHOW\_SPRITES, 313
- SPRITES\_8x16, 313
- SPRITES\_8x8, 313
- SWITCH\_RAM, 312
- SWITCH\_ROM, 312
- SWITCH\_ROM\_DUMMY, 311
- SWITCH\_ROM\_UNROM, 311
- sys\_time, 340
- SYSTEM\_50HZ, 306
- SYSTEM\_60HZ, 306
- SYSTEM\_BITS\_DENDY, 306
- SYSTEM\_BITS\_NTSC, 306
- SYSTEM\_BITS\_PAL, 306
- memset, 339
- vsync, 319
- wait\_vbl\_done, 319
- waitpad, 317
- waitpadup, 317
- NINTENDO
  - gb.h, 143
- NINTENDO\_NES
  - nes.h, 306
- NONBANKED
  - types.h, 113
- NORETURN
  - types.h, 113
- noreturn
  - stdnoreturn.h, 384
- nowait\_int\_handler
  - gb.h, 158
- npads
  - joypads\_t, 92
- NR10\_REG
  - hardware.h, 213
- NR11\_REG
  - hardware.h, 213
- NR12\_REG
  - hardware.h, 213
- NR13\_REG
  - hardware.h, 213
- NR14\_REG
  - hardware.h, 213
- NR21\_REG
  - hardware.h, 213
- NR22\_REG
  - hardware.h, 213
- NR23\_REG
  - hardware.h, 213
- NR24\_REG
  - hardware.h, 213
- NR30\_REG
  - hardware.h, 214
- NR31\_REG
  - hardware.h, 214
- NR32\_REG
  - hardware.h, 214
- NR33\_REG
  - hardware.h, 214
- NR34\_REG
  - hardware.h, 214
- NR41\_REG
  - hardware.h, 214
- NR42\_REG
  - hardware.h, 214
- NR43\_REG
  - hardware.h, 214
- NR44\_REG
  - hardware.h, 214
- NR50\_REG
  - hardware.h, 214
- NR51\_REG
  - hardware.h, 214
- NR52\_REG
  - hardware.h, 214
- NULL
  - stddef.h, 372
  - types.h, 115
- OAM\_item\_t, 93
  - gb.h, 155
  - msx.h, 287
  - nes.h, 314
  - prop, 94
  - tile, 94
  - x, 94
  - y, 94
- OAMF\_BANK0
  - hardware.h, 210
- OAMF\_BANK1
  - hardware.h, 210

OAMF\_CGB\_PAL0  
hardware.h, [210](#)

OAMF\_CGB\_PAL1  
hardware.h, [210](#)

OAMF\_CGB\_PAL2  
hardware.h, [210](#)

OAMF\_CGB\_PAL3  
hardware.h, [210](#)

OAMF\_CGB\_PAL4  
hardware.h, [210](#)

OAMF\_CGB\_PAL5  
hardware.h, [210](#)

OAMF\_CGB\_PAL6  
hardware.h, [210](#)

OAMF\_CGB\_PAL7  
hardware.h, [211](#)

OAMF\_PAL0  
hardware.h, [210](#)

OAMF\_PAL1  
hardware.h, [210](#)

OAMF\_PALMASK  
hardware.h, [211](#)

OAMF\_PRI  
hardware.h, [210](#)

OAMF\_XFLIP  
hardware.h, [210](#)

OAMF\_YFLIP  
hardware.h, [210](#)

OBP0\_REG  
hardware.h, [215](#)

OBP1\_REG  
hardware.h, [215](#)

OCPD\_REG  
hardware.h, [216](#)

OCPS\_REG  
hardware.h, [216](#)

OCPSF\_AUTOINC  
hardware.h, [208](#)

offsetof  
stddef.h, [373](#)

ofs  
\_\_far\_ptr, [89](#)

OLDCALL  
types.h, [112](#)

opcode  
isr\_nested\_vector\_t, [90](#)  
isr\_vector\_t, [91](#)

OR  
drawing.h, [132](#)

P1\_REG  
hardware.h, [212](#)

P1F\_0  
hardware.h, [199](#)

P1F\_1  
hardware.h, [199](#)

P1F\_2  
hardware.h, [199](#)

P1F\_3  
hardware.h, [199](#)

P1F\_4  
hardware.h, [199](#)

P1F\_5  
hardware.h, [199](#)

P1F\_GET\_BTN  
hardware.h, [199](#)

P1F\_GET\_DPAD  
hardware.h, [199](#)

P1F\_GET\_NONE  
hardware.h, [199](#)

palette\_color\_t  
cgb.h, [128](#)  
nes.h, [314](#)

PCM12\_REG  
hardware.h, [216](#)

PCM34\_REG  
hardware.h, [216](#)

PCM\_SAMPLE  
hardware.h, [214](#)

plot  
drawing.h, [134](#)

plot\_point  
drawing.h, [134](#)

pmfont\_handle  
font.h, [271](#)

POINTER  
types.h, [115](#)

posx  
console.h, [266](#)

posy  
console.h, [266](#)

PPUCTRL\_BG\_CHR  
hardware.h, [225](#)

PPUCTRL\_INC32  
hardware.h, [225](#)

PPUCTRL\_NMI  
hardware.h, [224](#)

PPUCTRL\_SPR\_8X16  
hardware.h, [224](#)

PPUCTRL\_SPR\_8X8  
hardware.h, [224](#)

PPUCTRL\_SPR\_CHR  
hardware.h, [225](#)

PPUMASK\_BLUE  
hardware.h, [225](#)

PPUMASK\_GREEN  
hardware.h, [225](#)

PPUMASK\_MONOCHROME  
hardware.h, [225](#)

PPUMASK\_RED  
hardware.h, [225](#)

PPUMASK\_SHOW\_BG  
hardware.h, [225](#)

PPUMASK\_SHOW\_BG\_LC  
hardware.h, [225](#)

PPUMASK\_SHOW\_SPR  
hardware.h, [225](#)

PPUMASK\_SHOW\_SPR\_LC  
     hardware.h, 225  
 PRESERVES\_REGS  
     types.h, 112  
 printf  
     stdio.h, 379  
 prop  
     OAM\_item\_t, 94  
 props  
     metasprite\_t, 93  
 provides.h  
     USE\_C\_MEMCPY, 95, 96  
     USE\_C\_STRCMP, 95, 96  
     USE\_C\_STRCPY, 95, 96  
 PSG\_CH0  
     hardware.h, 218, 233  
 PSG\_CH1  
     hardware.h, 218, 233  
 PSG\_CH2  
     hardware.h, 218, 234  
 PSG\_CH3  
     hardware.h, 218, 234  
 PSG\_LATCH  
     hardware.h, 218, 233  
 PSG\_VOLUME  
     hardware.h, 218, 234  
 ptr  
     \_\_far\_ptr, 89  
 PTRDIFF\_MAX  
     stdint.h, 376  
 PTRDIFF\_MIN  
     stdint.h, 376  
 ptrdiff\_t  
     stddef.h, 373  
 putchar  
     stdio.h, 379  
 puts  
     stdio.h, 380  
 qsort  
     stdlib.h, 383  
 R0\_CB\_INPUT  
     hardware.h, 219  
 R0\_CB\_OUTPUT  
     hardware.h, 219  
 R0\_DEFAULT  
     hardware.h, 219, 235  
 R0\_ES  
     hardware.h, 219, 235  
 R0\_ES\_OFF  
     hardware.h, 219, 235  
 R0\_HSCRL  
     hardware.h, 234  
 R0\_HSCRL\_INH  
     hardware.h, 234  
 R0\_IE1  
     hardware.h, 219, 234  
 R0\_IE1\_OFF  
     hardware.h, 219, 234  
 R0\_IE2  
     hardware.h, 219  
 R0\_IE2\_OFF  
     hardware.h, 219  
 R0\_LCB  
     hardware.h, 234  
 R0\_NO\_LCB  
     hardware.h, 234  
 R0\_SCR\_MODE1  
     hardware.h, 219  
 R0\_SCR\_MODE2  
     hardware.h, 219  
 R0\_SCR\_MODE3  
     hardware.h, 219  
 R0\_SS  
     hardware.h, 234  
 R0\_SS\_OFF  
     hardware.h, 234  
 R0\_VSCRL  
     hardware.h, 234  
 R0\_VSCRL\_INH  
     hardware.h, 234  
 R10\_INT\_EVERY  
     hardware.h, 222, 237  
 R10\_INT\_OFF  
     hardware.h, 222, 237  
 R1\_DEFAULT  
     hardware.h, 220, 235  
 R1\_DISP\_OFF  
     hardware.h, 220, 235  
 R1\_DISP\_ON  
     hardware.h, 220, 235  
 R1\_IE  
     hardware.h, 220, 235  
 R1\_IE\_OFF  
     hardware.h, 220, 235  
 R1\_SCR\_MODE1  
     hardware.h, 220  
 R1\_SCR\_MODE2  
     hardware.h, 220  
 R1\_SCR\_MODE3  
     hardware.h, 220  
 R1\_SPR\_16X16  
     hardware.h, 220  
 R1\_SPR\_8X16  
     hardware.h, 235  
 R1\_SPR\_8X8  
     hardware.h, 220, 235  
 R1\_SPR\_MAG  
     hardware.h, 220  
 R1\_SPR\_MAG\_OFF  
     hardware.h, 220  
 R2\_MAP\_0x0000  
     hardware.h, 221, 236  
 R2\_MAP\_0x0800  
     hardware.h, 221, 236  
 R2\_MAP\_0x1000

hardware.h, [221](#), [235](#)  
R2\_MAP\_0x1800  
hardware.h, [220](#), [235](#)  
R2\_MAP\_0x2000  
hardware.h, [220](#), [235](#)  
R2\_MAP\_0x2800  
hardware.h, [220](#), [235](#)  
R2\_MAP\_0x3000  
hardware.h, [220](#), [235](#)  
R2\_MAP\_0x3800  
hardware.h, [220](#), [235](#)  
R5\_SAT\_0x1F00  
hardware.h, [236](#)  
R5\_SAT\_0x3F00  
hardware.h, [221](#), [236](#)  
R5\_SAT\_MASK  
hardware.h, [221](#), [236](#)  
R6\_BANK0  
hardware.h, [221](#), [236](#)  
R6\_BANK1  
hardware.h, [221](#), [236](#)  
R6\_DATA\_0x0000  
hardware.h, [221](#), [236](#)  
R6\_DATA\_0x2000  
hardware.h, [221](#), [236](#)  
R7\_COLOR\_MASK  
hardware.h, [221](#), [236](#)  
RAMCTL\_BANK  
hardware.h, [238](#)  
RAMCTL\_PROT  
hardware.h, [239](#)  
RAMCTL\_RAM  
hardware.h, [239](#)  
RAMCTL\_RO  
hardware.h, [239](#)  
RAMCTL\_ROM  
hardware.h, [238](#)  
rand  
rand.h, [342](#)  
rand.h  
\_\_rand\_seed, [343](#)  
arand, [343](#)  
initarand, [343](#)  
initrand, [342](#)  
rand, [342](#)  
RAND\_MAX, [342](#)  
randw, [342](#)  
RANDW\_MAX, [342](#)  
RAND\_MAX  
rand.h, [342](#)  
randw  
rand.h, [342](#)  
RANDW\_MAX  
rand.h, [342](#)  
rAUD1ENV  
hardware.h, [201](#)  
rAUD1HIGH  
hardware.h, [201](#)  
rAUD1LEN  
hardware.h, [201](#)  
rAUD1LOW  
hardware.h, [201](#)  
rAUD1SWEEP  
hardware.h, [201](#)  
rAUD2ENV  
hardware.h, [201](#)  
rAUD2HIGH  
hardware.h, [201](#)  
rAUD2LEN  
hardware.h, [201](#)  
rAUD2LOW  
hardware.h, [201](#)  
rAUD3ENA  
hardware.h, [201](#)  
rAUD3HIGH  
hardware.h, [202](#)  
rAUD3LEN  
hardware.h, [202](#)  
rAUD3LEVEL  
hardware.h, [202](#)  
rAUD3LOW  
hardware.h, [202](#)  
rAUD4ENV  
hardware.h, [202](#)  
rAUD4GO  
hardware.h, [202](#)  
rAUD4LEN  
hardware.h, [202](#)  
rAUD4POLY  
hardware.h, [202](#)  
rAUDENA  
hardware.h, [203](#)  
rAUDTERM  
hardware.h, [202](#)  
rAUDVOL  
hardware.h, [202](#)  
rBCPD  
hardware.h, [208](#)  
rBCPS  
hardware.h, [208](#)  
rBGP  
hardware.h, [206](#)  
rDIV  
hardware.h, [200](#)  
rDMA  
hardware.h, [206](#)  
realloc  
stdlib.h, [383](#)  
receive\_byte  
gb.h, [159](#)  
refresh\_OAM  
gb.h, [162](#)  
msx.h, [290](#)  
nes.h, [320](#)  
sms.h, [358](#)  
remove\_JOY

- gb.h, [156](#)
- msx.h, [289](#)
- sms.h, [357](#)
- remove\_LCD
  - gb.h, [155](#)
  - msx.h, [289](#)
  - nes.h, [315](#)
  - sms.h, [357](#)
- remove\_SIO
  - gb.h, [156](#)
  - msx.h, [289](#)
  - sms.h, [357](#)
- remove\_TIM
  - gb.h, [155](#)
  - msx.h, [289](#)
  - sms.h, [357](#)
- remove\_VBL
  - gb.h, [155](#)
  - msx.h, [289](#)
  - nes.h, [315](#)
  - sms.h, [357](#)
- reset
  - gb.h, [162](#)
- RET\_SIZE
  - setjmp.h, [344](#)
- reverse
  - string.h, [100](#), [104](#), [108](#)
- RGB
  - cgb.h, [126](#)
  - nes.h, [307](#)
- RGB8
  - cgb.h, [126](#)
  - nes.h, [307](#)
- RGB\_AQUA
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_BLACK
  - cgb.h, [127](#)
  - nes.h, [308](#)
- RGB\_BLUE
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_BROWN
  - cgb.h, [128](#)
- RGB\_CYAN
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_DARKBLUE
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_DARKGRAY
  - cgb.h, [127](#)
  - nes.h, [308](#)
- RGB\_DARKGREEN
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_DARKRED
  - cgb.h, [127](#)
- nes.h, [307](#)
- RGB\_DARKYELLOW
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_GREEN
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_LIGHTFLESH
  - cgb.h, [127](#)
- RGB\_LIGHTGRAY
  - cgb.h, [127](#)
  - nes.h, [308](#)
- RGB\_ORANGE
  - cgb.h, [128](#)
- RGB\_PINK
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_PURPLE
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_RED
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGB\_TEAL
  - cgb.h, [128](#)
- RGB\_TO\_NES
  - rgb\_to\_nes\_macro.h, [341](#)
- rgb\_to\_nes\_macro.h
  - RGB\_TO\_NES, [341](#)
- RGB\_WHITE
  - cgb.h, [127](#)
  - nes.h, [308](#)
- RGB\_YELLOW
  - cgb.h, [127](#)
  - nes.h, [307](#)
- RGBHTML
  - cgb.h, [126](#)
  - nes.h, [307](#)
- rHDMA1
  - hardware.h, [207](#)
- rHDMA2
  - hardware.h, [207](#)
- rHDMA3
  - hardware.h, [208](#)
- rHDMA4
  - hardware.h, [208](#)
- rHDMA5
  - hardware.h, [208](#)
- rIE
  - hardware.h, [209](#)
- rIF
  - hardware.h, [201](#)
- rKEY1
  - hardware.h, [206](#)
- rLDC
  - hardware.h, [203](#)
- rle\_decompress
  - rledecompress.h, [274](#)

rle\_init  
    rledecompress.h, [274](#)  
RLE\_STOP  
    rledecompress.h, [274](#)  
rledecompress.h  
    rle\_decompress, [274](#)  
    rle\_init, [274](#)  
    RLE\_STOP, [274](#)  
rLY  
    hardware.h, [206](#)  
rLYC  
    hardware.h, [206](#)  
rOBP0  
    hardware.h, [206](#)  
rOBP1  
    hardware.h, [206](#)  
rOCPD  
    hardware.h, [208](#)  
rOCPS  
    hardware.h, [208](#)  
rP1  
    hardware.h, [199](#)  
RP\_REG  
    hardware.h, [216](#)  
rPCM12  
    hardware.h, [209](#)  
rPCM34  
    hardware.h, [209](#)  
RPF\_DATAIN  
    hardware.h, [208](#)  
RPF\_ENREAD  
    hardware.h, [208](#)  
RPF\_WRITE\_HI  
    hardware.h, [208](#)  
RPF\_WRITE\_LO  
    hardware.h, [208](#)  
rRAMB  
    hardware.h, [212](#)  
rRAMG  
    hardware.h, [212](#)  
rROMB0  
    hardware.h, [212](#)  
rROMB1  
    hardware.h, [212](#)  
rRP  
    hardware.h, [208](#)  
rSB  
    hardware.h, [199](#)  
rSC  
    hardware.h, [199](#)  
rSCX  
    hardware.h, [206](#)  
rSCY  
    hardware.h, [206](#)  
rSMBK  
    hardware.h, [209](#)  
rSPD  
    hardware.h, [206](#)  
rSTAT  
    hardware.h, [205](#)  
rSVBK  
    hardware.h, [208](#)  
rTAC  
    hardware.h, [200](#)  
rTIMA  
    hardware.h, [200](#)  
rTMA  
    hardware.h, [200](#)  
rVBK  
    hardware.h, [206](#)  
rWX  
    hardware.h, [206](#)  
rWY  
    hardware.h, [206](#)  
S\_BANK  
    gb.h, [145](#)  
    msx.h, [282](#)  
    sms.h, [349](#)  
S\_FLIPX  
    gb.h, [145](#)  
    msx.h, [282](#)  
    nes.h, [309](#)  
    sms.h, [350](#)  
S\_FLIPY  
    gb.h, [145](#)  
    msx.h, [282](#)  
    nes.h, [309](#)  
    sms.h, [350](#)  
S\_PAL  
    gb.h, [145](#)  
    msx.h, [282](#)  
    nes.h, [309](#)  
    sms.h, [350](#)  
S\_PALETTE  
    gb.h, [145](#)  
    msx.h, [282](#)  
    nes.h, [309](#)  
    sms.h, [350](#)  
S\_PRIORITY  
    gb.h, [145](#)  
    msx.h, [282](#)  
    nes.h, [309](#)  
    sms.h, [350](#)  
SB\_REG  
    hardware.h, [213](#)  
SC\_REG  
    hardware.h, [213](#)  
SCF\_SOURCE  
    hardware.h, [200](#)  
SCF\_SPEED  
    hardware.h, [200](#)  
SCF\_START  
    hardware.h, [200](#)  
SCHAR\_MAX  
    limits.h, [275](#)  
SCHAR\_MIN

limits.h, [276](#)  
SCREENHEIGHT  
  gb.h, [147](#)  
  msx.h, [283](#)  
  nes.h, [310](#)  
  sms.h, [351](#)  
SCREENWIDTH  
  gb.h, [147](#)  
  msx.h, [283](#)  
  nes.h, [310](#)  
  sms.h, [351](#)  
scroll\_bkg  
  gb.h, [173](#)  
  msx.h, [290](#)  
  nes.h, [330](#)  
  sms.h, [358](#)  
scroll\_sprite  
  gb.h, [183](#)  
  msx.h, [299](#)  
  nes.h, [334](#)  
  sms.h, [368](#)  
scroll\_win  
  gb.h, [179](#)  
SCX\_REG  
  hardware.h, [214](#)  
SCY\_REG  
  hardware.h, [214](#)  
seg  
  \_\_far\_ptr, [89](#)  
SEGA  
  sms.h, [348](#)  
segn  
  \_\_far\_ptr, [89](#)  
segofs  
  \_\_far\_ptr, [89](#)  
send\_byte  
  gb.h, [159](#)  
set\_1bpp\_colors  
  gb.h, [163](#)  
  msx.h, [293](#)  
  nes.h, [320](#)  
  sms.h, [362](#)  
set\_1bpp\_colors\_ex  
  gb.h, [163](#)  
  nes.h, [320](#)  
set\_2bpp\_palette  
  gb.h, [163](#)  
  msx.h, [293](#)  
  nes.h, [320](#)  
  sms.h, [362](#)  
set\_attribute\_xy  
  gb.h, [154](#)  
  nes.h, [313](#)  
  sms.h, [369](#)  
set\_attributed\_tile\_xy  
  msx.h, [300](#)  
  sms.h, [369](#)  
set\_bkg\_1bpp\_data  
  gb.h, [164](#)  
  msx.h, [293](#)  
  nes.h, [321](#)  
  sms.h, [362](#)  
set\_bkg\_2bpp\_data  
  gb.h, [154](#)  
  nes.h, [313](#)  
  sms.h, [362](#)  
set\_bkg\_4bpp\_data  
  msx.h, [293](#)  
  sms.h, [361](#)  
set\_bkg\_attribute\_xy  
  gb.h, [172](#)  
  nes.h, [328](#)  
  sms.h, [355](#)  
set\_bkg\_attribute\_xy\_nes16x16  
  nes.h, [328](#)  
set\_bkg\_attributes  
  gb.h, [168](#)  
  nes.h, [324](#)  
  sms.h, [364](#)  
set\_bkg\_attributes\_nes16x16  
  nes.h, [323](#)  
set\_bkg\_based\_submap  
  gb.h, [170](#)  
  msx.h, [296](#)  
  nes.h, [327](#)  
  sms.h, [366](#)  
set\_bkg\_based\_tiles  
  gb.h, [167](#)  
  msx.h, [294](#)  
  nes.h, [325](#)  
  sms.h, [363](#)  
set\_bkg\_data  
  gb.h, [164](#)  
  msx.h, [293](#)  
  nes.h, [320](#)  
  sms.h, [362](#)  
set\_bkg\_native\_data  
  gb.h, [186](#)  
  nes.h, [336](#)  
  sms.h, [361](#)  
set\_bkg\_palette  
  cgb.h, [128](#)  
  msx.h, [286](#)  
  nes.h, [314](#)  
  sms.h, [354](#)  
set\_bkg\_palette\_entry  
  cgb.h, [129](#)  
  msx.h, [286](#)  
  nes.h, [315](#)  
  sms.h, [354](#)  
set\_bkg\_submap  
  gb.h, [169](#)  
  msx.h, [295](#)  
  nes.h, [326](#)  
  sms.h, [364](#)  
set\_bkg\_submap\_attributes



- gb.h, [170](#)
- nes.h, [325](#)
- sms.h, [366](#)
- set\_bkg\_submap\_attributes\_nes16x16
  - nes.h, [324](#)
- set\_bkg\_tile\_xy
  - gb.h, [172](#)
  - msx.h, [287](#)
  - nes.h, [328](#)
  - sms.h, [355](#)
- set\_bkg\_tiles
  - gb.h, [166](#)
  - msx.h, [287](#)
  - nes.h, [322](#)
  - sms.h, [355](#)
- SET\_BORDER\_COLOR
  - gb.h, [153](#)
  - msx.h, [284](#)
  - nes.h, [313](#)
  - sms.h, [352](#)
- set\_data
  - gb.h, [183](#)
  - msx.h, [294](#)
  - nes.h, [336](#)
  - sms.h, [363](#)
- set\_default\_palette
  - cgb.h, [130](#)
  - msx.h, [292](#)
  - sms.h, [360](#)
- set\_interrupts
  - gb.h, [161](#)
  - msx.h, [288](#)
  - sms.h, [356](#)
- set\_native\_sprite\_data
  - msx.h, [293](#)
- set\_native\_tile\_data
  - gb.h, [186](#)
  - msx.h, [292](#)
  - nes.h, [339](#)
  - sms.h, [361](#)
- set\_palette
  - msx.h, [292](#)
  - sms.h, [360](#)
- set\_palette\_entry
  - msx.h, [292](#)
  - sms.h, [360](#)
- SET\_SHADOW\_OAM\_ADDRESS
  - gb.h, [180](#)
  - msx.h, [297](#)
  - nes.h, [331](#)
  - sms.h, [366](#)
- set\_sprite\_1bpp\_data
  - gb.h, [180](#)
  - msx.h, [293](#)
  - nes.h, [331](#)
  - sms.h, [363](#)
- set\_sprite\_2bpp\_data
  - gb.h, [154](#)
- nes.h, [313](#)
- sms.h, [362](#)
- set\_sprite\_4bpp\_data
  - sms.h, [361](#)
- set\_sprite\_data
  - gb.h, [179](#)
  - msx.h, [293](#)
  - nes.h, [330](#)
  - sms.h, [362](#)
- set\_sprite\_native\_data
  - gb.h, [187](#)
  - nes.h, [338](#)
  - sms.h, [361](#)
- set\_sprite\_palette
  - cgb.h, [128](#)
  - msx.h, [286](#)
  - nes.h, [314](#)
  - sms.h, [355](#)
- set\_sprite\_palette\_entry
  - cgb.h, [129](#)
  - msx.h, [286](#)
  - nes.h, [315](#)
  - sms.h, [354](#)
- set\_sprite\_prop
  - gb.h, [181](#)
  - msx.h, [297](#)
  - nes.h, [332](#)
  - sms.h, [367](#)
- set\_sprite\_tile
  - gb.h, [181](#)
  - msx.h, [297](#)
  - nes.h, [331](#)
  - sms.h, [366](#)
- set\_tile\_1bpp\_data
  - msx.h, [293](#)
  - sms.h, [362](#)
- set\_tile\_2bpp\_data
  - sms.h, [362](#)
- set\_tile\_data
  - gb.h, [185](#)
  - nes.h, [336](#)
- set\_tile\_map
  - gb.h, [154](#)
  - msx.h, [294](#)
  - nes.h, [313](#)
  - sms.h, [363](#)
- set\_tile\_map\_compat
  - sms.h, [363](#)
- set\_tile\_submap
  - gb.h, [154](#)
  - msx.h, [294](#)
  - nes.h, [313](#)
  - sms.h, [364](#)
- set\_tile\_submap\_compat
  - msx.h, [294](#)
  - sms.h, [364](#)
- set\_tile\_xy
  - gb.h, [154](#)

- msx.h, [300](#)
- nes.h, [313](#)
- sms.h, [369](#)
- set\_tiles
  - gb.h, [184](#)
  - nes.h, [336](#)
- set\_vram\_byte
  - gb.h, [163](#)
  - msx.h, [300](#)
  - nes.h, [320](#)
  - sms.h, [368](#)
- set\_win\_1bpp\_data
  - gb.h, [174](#)
- set\_win\_attribute\_xy
  - sms.h, [355](#)
- set\_win\_based\_submap
  - gb.h, [177](#)
  - msx.h, [296](#)
  - sms.h, [366](#)
- set\_win\_based\_tiles
  - gb.h, [175](#)
  - msx.h, [294](#)
  - sms.h, [363](#)
- set\_win\_data
  - gb.h, [173](#)
- set\_win\_submap
  - gb.h, [176](#)
  - msx.h, [295](#)
  - sms.h, [365](#)
- set\_win\_tile\_xy
  - gb.h, [178](#)
  - msx.h, [287](#)
  - sms.h, [355](#)
- set\_win\_tiles
  - gb.h, [175](#)
  - msx.h, [287](#)
  - sms.h, [355](#)
- setchar
  - console.h, [267](#)
- setjmp
  - setjmp.h, [344](#)
- setjmp.h
  - \_\_setjmp, [344](#)
  - BP\_SIZE, [344](#)
  - BPX\_SIZE, [344](#)
  - jmp\_buf, [344](#)
  - longjmp, [344](#)
  - RET\_SIZE, [344](#)
  - setjmp, [344](#)
  - SP\_SIZE, [343](#)
  - SPX\_SIZE, [344](#)
- sfont\_handle, [94](#)
  - first\_tile, [95](#)
  - font, [95](#)
- SFR
  - types.h, [113](#)
- sgb.h
  - c, [266](#)
  - SGB\_ATTRC\_EN, [264](#)
  - SGB\_ATTR\_BLK, [264](#)
  - SGB\_ATTR\_CHR, [264](#)
  - SGB\_ATTR\_DIV, [264](#)
  - SGB\_ATTR\_LIN, [264](#)
  - SGB\_ATTR\_SET, [265](#)
  - SGB\_ATTR\_TRN, [265](#)
  - sgb\_check, [265](#)
  - SGB\_CHR\_TRN, [265](#)
  - SGB\_DATA\_SND, [265](#)
  - SGB\_DATA\_TRN, [265](#)
  - SGB\_ICON\_EN, [264](#)
  - SGB\_JUMP, [265](#)
  - SGB\_MASK\_EN, [265](#)
  - SGB\_MLT\_REQ, [265](#)
  - SGB\_OBJ\_TRN, [265](#)
  - SGB\_PAL\_01, [264](#)
  - SGB\_PAL\_03, [264](#)
  - SGB\_PAL\_12, [264](#)
  - SGB\_PAL\_23, [264](#)
  - SGB\_PAL\_SET, [264](#)
  - SGB\_PAL\_TRN, [264](#)
  - SGB\_PCT\_TRN, [265](#)
  - SGB\_SOU\_TRN, [264](#)
  - SGB\_SOUND, [264](#)
  - SGB\_TEST\_EN, [264](#)
  - sgb\_transfer, [265](#)
- SGB\_ATTRC\_EN
  - sgb.h, [264](#)
- SGB\_ATTR\_BLK
  - sgb.h, [264](#)
- SGB\_ATTR\_CHR
  - sgb.h, [264](#)
- SGB\_ATTR\_DIV
  - sgb.h, [264](#)
- SGB\_ATTR\_LIN
  - sgb.h, [264](#)
- SGB\_ATTR\_SET
  - sgb.h, [265](#)
- SGB\_ATTR\_TRN
  - sgb.h, [265](#)
- sgb\_check
  - sgb.h, [265](#)
- SGB\_CHR\_TRN
  - sgb.h, [265](#)
- SGB\_DATA\_SND
  - sgb.h, [265](#)
- SGB\_DATA\_TRN
  - sgb.h, [265](#)
- SGB\_ICON\_EN
  - sgb.h, [264](#)
- SGB\_JUMP
  - sgb.h, [265](#)
- SGB\_MASK\_EN
  - sgb.h, [265](#)
- SGB\_MLT\_REQ
  - sgb.h, [265](#)
- SGB\_OBJ\_TRN

- sgb.h, [265](#)
- SGB\_PAL\_01
  - sgb.h, [264](#)
- SGB\_PAL\_03
  - sgb.h, [264](#)
- SGB\_PAL\_12
  - sgb.h, [264](#)
- SGB\_PAL\_23
  - sgb.h, [264](#)
- SGB\_PAL\_SET
  - sgb.h, [264](#)
- SGB\_PAL\_TRN
  - sgb.h, [264](#)
- SGB\_PCT\_TRN
  - sgb.h, [265](#)
- SGB\_SOU\_TRN
  - sgb.h, [264](#)
- SGB\_SOUND
  - sgb.h, [264](#)
- SGB\_TEST\_EN
  - sgb.h, [264](#)
- sgb\_transfer
  - sgb.h, [265](#)
- shadow\_OAM
  - gb.h, [190](#)
  - msx.h, [302](#)
  - nes.h, [341](#)
  - sms.h, [371](#)
- shadow\_PPUCTRL
  - hardware.h, [227](#)
- shadow\_PPUMASK
  - hardware.h, [227](#)
- shadow\_VDP\_R0
  - hardware.h, [222](#), [239](#)
- shadow\_VDP\_R1
  - hardware.h, [222](#), [239](#)
- shadow\_VDP\_R10
  - hardware.h, [223](#), [240](#)
- shadow\_VDP\_R2
  - hardware.h, [222](#), [239](#)
- shadow\_VDP\_R3
  - hardware.h, [222](#), [239](#)
- shadow\_VDP\_R4
  - hardware.h, [222](#), [239](#)
- shadow\_VDP\_R5
  - hardware.h, [223](#), [239](#)
- shadow\_VDP\_R6
  - hardware.h, [223](#), [239](#)
- shadow\_VDP\_R7
  - hardware.h, [223](#), [239](#)
- shadow\_VDP\_R8
  - hardware.h, [223](#), [240](#)
- shadow\_VDP\_R9
  - hardware.h, [223](#), [240](#)
- shadow\_VDP\_RBORDER
  - hardware.h, [223](#), [239](#)
- shadow\_VDP\_RSCX
  - hardware.h, [223](#), [240](#)
- shadow\_VDP\_RSCY
  - hardware.h, [223](#), [240](#)
- SHOW\_BKG
  - gb.h, [153](#)
  - msx.h, [284](#)
  - nes.h, [313](#)
  - sms.h, [352](#)
- SHOW\_LEFT\_COLUMN
  - gb.h, [153](#)
  - msx.h, [284](#)
  - nes.h, [312](#)
  - sms.h, [352](#)
- SHOW\_SPRITES
  - gb.h, [153](#)
  - msx.h, [284](#)
  - nes.h, [313](#)
  - sms.h, [352](#)
- SHOW\_WIN
  - gb.h, [153](#)
  - msx.h, [284](#)
  - sms.h, [352](#)
- SHRT\_MAX
  - limits.h, [276](#)
- SHRT\_MIN
  - limits.h, [276](#)
- SIG\_ATOMIC\_MAX
  - stdint.h, [376](#)
- SIG\_ATOMIC\_MIN
  - stdint.h, [376](#)
- SIGNED
  - drawing.h, [132](#)
- SIO\_IFLAG
  - gb.h, [146](#)
  - msx.h, [283](#)
  - sms.h, [351](#)
- SIOCTL\_BS0
  - hardware.h, [231](#)
- SIOCTL\_BS1
  - hardware.h, [231](#)
- SIOCTL\_FRER
  - hardware.h, [231](#)
- SIOCTL\_INT
  - hardware.h, [231](#)
- SIOCTL\_RON
  - hardware.h, [231](#)
- SIOCTL\_RXRD
  - hardware.h, [231](#)
- SIOCTL\_TON
  - hardware.h, [231](#)
- SIOCTL\_TXFL
  - hardware.h, [231](#)
- SIOF\_B\_CLOCK
  - hardware.h, [200](#)
- SIOF\_B\_SPEED
  - hardware.h, [200](#)
- SIOF\_B\_XFER\_START
  - hardware.h, [200](#)
- SIOF\_CLOCK\_EXT

- hardware.h, [200](#)
- SIOF\_CLOCK\_INT
  - hardware.h, [200](#)
- SIOF\_SPEED\_1X
  - hardware.h, [200](#)
- SIOF\_SPEED\_32X
  - hardware.h, [200](#)
- SIOF\_XFER\_START
  - hardware.h, [199](#)
- SIZE\_MAX
  - stdint.h, [376](#)
- size\_t
  - stddef.h, [373](#)
  - types.h, [111](#), [112](#), [115](#)
- sms.h
  - \_BIOS, [370](#)
  - \_SYSTEM, [370](#)
  - \_\_READ\_VDP\_REG, [350](#)
  - \_\_WRITE\_VDP\_REG, [350](#)
  - \_\_WRITE\_VDP\_REG\_UNSAFE, [350](#)
  - \_current\_1bpp\_colors, [370](#)
  - \_current\_2bpp\_palette, [370](#)
  - \_current\_bank, [353](#)
  - \_map\_tile\_offset, [370](#)
  - \_shadow\_OAM\_OFF, [371](#)
  - \_shadow\_OAM\_base, [371](#)
  - \_sprites\_OFF, [371](#)
  - \_submap\_tile\_offset, [370](#)
  - add\_JOY, [358](#)
  - add\_LCD, [357](#)
  - add\_SIO, [357](#)
  - add\_TIM, [357](#)
  - add\_VBL, [357](#)
  - b, [370](#)
  - BANK, [353](#)
  - BANKREF, [353](#)
  - BANKREF\_EXTERN, [353](#)
  - c, [370](#)
  - cancel\_pending\_interrupts, [358](#)
  - cgb\_compatibility, [360](#)
  - COMPAT\_PALETTE, [355](#)
  - cpu\_fast, [360](#)
  - CURRENT\_BANK, [353](#)
  - d, [370](#)
  - delay, [358](#)
  - DEVICE\_SUPPORTS\_COLOR, [352](#)
  - disable\_interrupts, [360](#)
  - DISABLE\_RAM, [354](#)
  - DISABLE\_VBL\_TRANSFER, [355](#)
  - DISPLAY\_OFF, [351](#)
  - display\_off, [358](#)
  - DISPLAY\_ON, [351](#)
  - DIV\_REG, [353](#)
  - e, [370](#)
  - EMPTY\_IFLAG, [350](#)
  - enable\_interrupts, [359](#)
  - ENABLE\_RAM, [354](#)
  - ENABLE\_VBL\_TRANSFER, [355](#)
  - fill\_bkg\_rect, [355](#)
  - fill\_rect, [366](#)
  - fill\_rect\_compat, [366](#)
  - fill\_win\_rect, [355](#)
  - get\_bkg\_xy\_addr, [369](#)
  - get\_mode, [356](#)
  - get\_r\_reg, [358](#)
  - get\_sprite\_prop, [367](#)
  - get\_sprite\_tile, [367](#)
  - get\_system, [356](#)
  - get\_win\_xy\_addr, [356](#)
  - h, [370](#)
  - HARDWARE\_SPRITE\_CAN\_FLIP\_X, [355](#)
  - HARDWARE\_SPRITE\_CAN\_FLIP\_Y, [355](#)
  - HIDE\_BKG, [352](#)
  - HIDE\_LEFT\_COLUMN, [352](#)
  - hide\_sprite, [368](#)
  - HIDE\_SPRITES, [352](#)
  - HIDE\_WIN, [352](#)
  - int\_handler, [356](#)
  - iyh, [370](#)
  - iyl, [370](#)
  - J\_A, [349](#)
  - J\_B, [349](#)
  - J\_DOWN, [349](#)
  - J\_LEFT, [349](#)
  - J\_RIGHT, [349](#)
  - J\_SELECT, [349](#)
  - J\_START, [349](#)
  - J\_UP, [348](#)
  - JOY\_IFLAG, [351](#)
  - joypad, [358](#)
  - joypad\_ex, [359](#)
  - joypad\_init, [359](#)
  - l, [370](#)
  - LCD\_IFLAG, [350](#)
  - M\_NO\_INTERP, [349](#)
  - M\_NO\_SCROLL, [349](#)
  - M\_TEXT\_INOUT, [349](#)
  - M\_TEXT\_OUT, [349](#)
  - MAX\_HARDWARE\_SPRITES, [355](#)
  - MAXWNDPOSX, [351](#)
  - MAXWNDPOSY, [351](#)
  - MINWNDPOSX, [351](#)
  - MINWNDPOSY, [351](#)
  - mode, [356](#)
  - move\_bkg, [358](#)
  - move\_sprite, [367](#)
  - refresh\_OAM, [358](#)
  - remove\_JOY, [357](#)
  - remove\_LCD, [357](#)
  - remove\_SIO, [357](#)
  - remove\_TIM, [357](#)
  - remove\_VBL, [357](#)
  - S\_BANK, [349](#)
  - S\_FLIPX, [350](#)
  - S\_FLIPY, [350](#)
  - S\_PAL, [350](#)

S\_PALETTE, [350](#)  
S\_PRIORITY, [350](#)  
SCREENHEIGHT, [351](#)  
SCREENWIDTH, [351](#)  
scroll\_bkg, [358](#)  
scroll\_sprite, [368](#)  
SEGA, [348](#)  
set\_1bpp\_colors, [362](#)  
set\_2bpp\_palette, [362](#)  
set\_attribute\_xy, [369](#)  
set\_attributed\_tile\_xy, [369](#)  
set\_bkg\_1bpp\_data, [362](#)  
set\_bkg\_2bpp\_data, [362](#)  
set\_bkg\_4bpp\_data, [361](#)  
set\_bkg\_attribute\_xy, [355](#)  
set\_bkg\_attributes, [364](#)  
set\_bkg\_based\_submap, [366](#)  
set\_bkg\_based\_tiles, [363](#)  
set\_bkg\_data, [362](#)  
set\_bkg\_native\_data, [361](#)  
set\_bkg\_palette, [354](#)  
set\_bkg\_palette\_entry, [354](#)  
set\_bkg\_submap, [364](#)  
set\_bkg\_submap\_attributes, [366](#)  
set\_bkg\_tile\_xy, [355](#)  
set\_bkg\_tiles, [355](#)  
SET\_BORDER\_COLOR, [352](#)  
set\_data, [363](#)  
set\_default\_palette, [360](#)  
set\_interrupts, [356](#)  
set\_native\_tile\_data, [361](#)  
set\_palette, [360](#)  
set\_palette\_entry, [360](#)  
SET\_SHADOW\_OAM\_ADDRESS, [366](#)  
set\_sprite\_1bpp\_data, [363](#)  
set\_sprite\_2bpp\_data, [362](#)  
set\_sprite\_4bpp\_data, [361](#)  
set\_sprite\_data, [362](#)  
set\_sprite\_native\_data, [361](#)  
set\_sprite\_palette, [355](#)  
set\_sprite\_palette\_entry, [354](#)  
set\_sprite\_prop, [367](#)  
set\_sprite\_tile, [366](#)  
set\_tile\_1bpp\_data, [362](#)  
set\_tile\_2bpp\_data, [362](#)  
set\_tile\_map, [363](#)  
set\_tile\_map\_compat, [363](#)  
set\_tile\_submap, [364](#)  
set\_tile\_submap\_compat, [364](#)  
set\_tile\_xy, [369](#)  
set\_vram\_byte, [368](#)  
set\_win\_attribute\_xy, [355](#)  
set\_win\_based\_submap, [366](#)  
set\_win\_based\_tiles, [363](#)  
set\_win\_submap, [365](#)  
set\_win\_tile\_xy, [355](#)  
set\_win\_tiles, [355](#)  
shadow\_OAM, [371](#)  
  
SHOW\_BKG, [352](#)  
SHOW\_LEFT\_COLUMN, [352](#)  
SHOW\_SPRITES, [352](#)  
SHOW\_WIN, [352](#)  
SIO\_IFLAG, [351](#)  
SPRITES\_8x16, [352](#)  
SPRITES\_8x8, [352](#)  
SWITCH\_RAM, [354](#)  
SWITCH\_ROM, [354](#)  
SWITCH\_ROM1, [354](#)  
SWITCH\_ROM2, [354](#)  
sys\_time, [370](#)  
SYSTEM\_50HZ, [348](#)  
SYSTEM\_60HZ, [348](#)  
TIM\_IFLAG, [350](#)  
VBK\_REG, [348](#)  
VBL\_IFLAG, [350](#)  
vmemcpy, [363](#)  
vsync, [358](#)  
wait\_vbl\_done, [358](#)  
waitpad, [359](#)  
waitpadup, [359](#)  
WRITE\_VDP\_CMD, [356](#)  
WRITE\_VDP\_DATA, [356](#)  
  
SOLID  
    drawing.h, [132](#)  
SOUNDPAN\_NOSL  
    hardware.h, [231](#)  
SOUNDPAN\_NOSR  
    hardware.h, [231](#)  
SOUNDPAN\_TN1L  
    hardware.h, [231](#)  
SOUNDPAN\_TN1R  
    hardware.h, [231](#)  
SOUNDPAN\_TN2L  
    hardware.h, [231](#)  
SOUNDPAN\_TN2R  
    hardware.h, [231](#)  
SOUNDPAN\_TN3L  
    hardware.h, [231](#)  
SOUNDPAN\_TN3R  
    hardware.h, [231](#)  
SP\_SIZE  
    setjmp.h, [343](#)  
sprintf  
    stdio.h, [380](#)  
SPRITES\_16x16  
    msx.h, [284](#)  
SPRITES\_8x16  
    gb.h, [154](#)  
    nes.h, [313](#)  
    sms.h, [352](#)  
SPRITES\_8x8  
    gb.h, [154](#)  
    msx.h, [285](#)  
    nes.h, [313](#)  
    sms.h, [352](#)  
SPX\_SIZE

- setjmp.h, [344](#)
- STAT\_REG
  - hardware.h, [214](#)
- STATF\_9\_SPR
  - hardware.h, [219](#), [234](#)
- STATF\_B\_BUSY
  - hardware.h, [206](#)
- STATF\_B\_LYC
  - hardware.h, [205](#)
- STATF\_B\_LYCF
  - hardware.h, [205](#)
- STATF\_B\_MODE00
  - hardware.h, [205](#)
- STATF\_B\_MODE01
  - hardware.h, [205](#)
- STATF\_B\_MODE10
  - hardware.h, [205](#)
- STATF\_B\_OAM
  - hardware.h, [206](#)
- STATF\_B\_VBL
  - hardware.h, [205](#)
- STATF\_BUSY
  - hardware.h, [205](#)
- STATF\_HBL
  - hardware.h, [205](#)
- STATF\_INT\_VBL
  - hardware.h, [219](#), [234](#)
- STATF\_LCD
  - hardware.h, [205](#)
- STATF\_LYC
  - hardware.h, [205](#)
- STATF\_LYCF
  - hardware.h, [205](#)
- STATF\_MODE00
  - hardware.h, [205](#)
- STATF\_MODE01
  - hardware.h, [205](#)
- STATF\_MODE10
  - hardware.h, [205](#)
- STATF\_OAM
  - hardware.h, [205](#)
- STATF\_SPR\_COLL
  - hardware.h, [219](#), [234](#)
- STATF\_VBL
  - hardware.h, [205](#)
- stdarg.h
  - va\_arg, [97](#), [98](#)
  - va\_end, [97](#), [98](#)
  - va\_list, [97](#), [98](#)
  - va\_start, [96–98](#)
- stdatomic.h
  - atomic\_flag\_clear, [371](#)
  - atomic\_flag\_test\_and\_set, [371](#)
- stdbool.h
  - \_\_bool\_true\_false\_are\_defined, [372](#)
  - bool, [372](#)
  - false, [372](#)
  - true, [372](#)
- stddef.h
  - \_\_PTRDIFF\_T\_DEFINED, [372](#)
  - \_\_SIZE\_T\_DEFINED, [372](#)
  - \_\_WCHAR\_T\_DEFINED, [372](#)
  - NULL, [372](#)
  - offsetof, [373](#)
  - ptrdiff\_t, [373](#)
  - size\_t, [373](#)
  - wchar\_t, [373](#)
- stdint.h
  - INT16\_C, [376](#)
  - INT16\_MAX, [375](#)
  - INT16\_MIN, [374](#)
  - int16\_t, [377](#)
  - INT32\_C, [377](#)
  - INT32\_MAX, [375](#)
  - INT32\_MIN, [374](#)
  - int32\_t, [377](#)
  - INT8\_C, [376](#)
  - INT8\_MAX, [374](#)
  - INT8\_MIN, [374](#)
  - int8\_t, [377](#)
  - INT\_FAST16\_MAX, [376](#)
  - INT\_FAST16\_MIN, [375](#)
  - int\_fast16\_t, [378](#)
  - INT\_FAST32\_MAX, [376](#)
  - INT\_FAST32\_MIN, [375](#)
  - int\_fast32\_t, [378](#)
  - INT\_FAST8\_MAX, [375](#)
  - INT\_FAST8\_MIN, [375](#)
  - int\_fast8\_t, [378](#)
  - INT\_LEAST16\_MAX, [375](#)
  - INT\_LEAST16\_MIN, [375](#)
  - int\_least16\_t, [378](#)
  - INT\_LEAST32\_MAX, [375](#)
  - INT\_LEAST32\_MIN, [375](#)
  - int\_least32\_t, [378](#)
  - INT\_LEAST8\_MAX, [375](#)
  - INT\_LEAST8\_MIN, [375](#)
  - int\_least8\_t, [378](#)
  - INTMAX\_C, [377](#)
  - INTMAX\_MAX, [376](#)
  - INTMAX\_MIN, [376](#)
  - intmax\_t, [378](#)
  - INTPTR\_MAX, [376](#)
  - INTPTR\_MIN, [376](#)
  - intptr\_t, [378](#)
  - PTRDIFF\_MAX, [376](#)
  - PTRDIFF\_MIN, [376](#)
  - SIG\_ATOMIC\_MAX, [376](#)
  - SIG\_ATOMIC\_MIN, [376](#)
  - SIZE\_MAX, [376](#)
  - UINT16\_C, [377](#)
  - UINT16\_MAX, [375](#)
  - uint16\_t, [377](#)
  - UINT32\_C, [377](#)
  - UINT32\_MAX, [375](#)
  - uint32\_t, [378](#)

- UINT8\_C, [377](#)
- UINT8\_MAX, [375](#)
- uint8\_t, [377](#)
- UINT\_FAST16\_MAX, [376](#)
- uint\_fast16\_t, [378](#)
- UINT\_FAST32\_MAX, [376](#)
- uint\_fast32\_t, [378](#)
- UINT\_FAST8\_MAX, [376](#)
- uint\_fast8\_t, [378](#)
- UINT\_LEAST16\_MAX, [375](#)
- uint\_least16\_t, [378](#)
- UINT\_LEAST32\_MAX, [375](#)
- uint\_least32\_t, [378](#)
- UINT\_LEAST8\_MAX, [375](#)
- uint\_least8\_t, [378](#)
- UINTMAX\_C, [377](#)
- UINTMAX\_MAX, [376](#)
- uintmax\_t, [378](#)
- UINTPTR\_MAX, [376](#)
- uintptr\_t, [378](#)
- WCHAR\_MAX, [377](#)
- WCHAR\_MIN, [377](#)
- WINT\_MAX, [377](#)
- WINT\_MIN, [377](#)
- stdio.h
  - getchar, [380](#)
  - gets, [380](#)
  - printf, [379](#)
  - putchar, [379](#)
  - puts, [380](#)
  - sprintf, [380](#)
- stdlib.h
  - abs, [381](#)
  - atoi, [381](#)
  - atol, [381](#)
  - bsearch, [383](#)
  - calloc, [383](#)
  - exit, [381](#)
  - free, [383](#)
  - itoa, [382](#)
  - labs, [381](#)
  - ltoa, [382](#)
  - malloc, [383](#)
  - qsort, [383](#)
  - realloc, [383](#)
  - uitoa, [382](#)
  - ultoa, [382](#)
- stdnoreturn.h
  - noreturn, [384](#)
- strcat
  - string.h, [100](#), [104](#), [108](#)
- strcmp
  - string.h, [99](#), [103](#), [107](#)
- strcpy
  - string.h, [99](#), [103](#), [106](#)
- string.h
  - \_\_memcpy, [99](#)
  - c, [106](#)
  - memcmp, [102](#), [105](#), [109](#)
  - memcpy, [99](#), [103](#), [107](#)
  - memmove, [100](#), [103](#), [107](#)
  - memset, [100](#), [104](#), [107](#)
  - reverse, [100](#), [104](#), [108](#)
  - strcat, [100](#), [104](#), [108](#)
  - strcmp, [99](#), [103](#), [107](#)
  - strcpy, [99](#), [103](#), [106](#)
  - strlen, [101](#), [104](#), [108](#)
  - strncat, [101](#), [104](#), [108](#)
  - strncmp, [101](#), [105](#), [109](#)
  - strncpy, [101](#), [105](#), [109](#)
- strlen
  - string.h, [101](#), [104](#), [108](#)
- strncat
  - string.h, [101](#), [104](#), [108](#)
- strncmp
  - string.h, [101](#), [105](#), [109](#)
- strncpy
  - string.h, [101](#), [105](#), [109](#)
- SVBK\_REG
  - hardware.h, [216](#)
- SWITCH\_16\_8\_MODE\_MBC1
  - gb.h, [151](#)
- SWITCH\_4\_32\_MODE\_MBC1
  - gb.h, [151](#)
- switch\_data
  - drawing.h, [134](#)
- SWITCH\_RAM
  - gb.h, [150](#)
  - msx.h, [286](#)
  - nes.h, [312](#)
  - sms.h, [354](#)
- SWITCH\_RAM\_MBC1
  - gb.h, [151](#)
- SWITCH\_RAM\_MBC5
  - gb.h, [152](#)
- SWITCH\_ROM
  - gb.h, [150](#)
  - msx.h, [290](#)
  - nes.h, [312](#)
  - sms.h, [354](#)
- SWITCH\_ROM1
  - msx.h, [286](#)
  - sms.h, [354](#)
- SWITCH\_ROM2
  - msx.h, [286](#)
  - sms.h, [354](#)
- SWITCH\_ROM\_DUMMY
  - nes.h, [311](#)
- SWITCH\_ROM\_MBC1
  - gb.h, [151](#)
- SWITCH\_ROM\_MBC5
  - gb.h, [151](#)
- SWITCH\_ROM\_MBC5\_8M
  - gb.h, [152](#)
- SWITCH\_ROM\_MEGADUCK
  - gb.h, [151](#)

SWITCH\_ROM\_UNROM  
     nes.h, [311](#)  
 sys\_time  
     gb.h, [189](#)  
     msx.h, [301](#)  
     nes.h, [340](#)  
     sms.h, [370](#)  
 SYSTEM\_50HZ  
     gb.h, [144](#)  
     msx.h, [280](#)  
     nes.h, [306](#)  
     sms.h, [348](#)  
 SYSTEM\_60HZ  
     gb.h, [143](#)  
     msx.h, [280](#)  
     nes.h, [306](#)  
     sms.h, [348](#)  
 SYSTEM\_BITS\_DENDY  
     nes.h, [306](#)  
 SYSTEM\_BITS\_NTSC  
     nes.h, [306](#)  
 SYSTEM\_BITS\_PAL  
     nes.h, [306](#)  
 SYSTEM\_NTSC  
     hardware.h, [222](#)  
 SYSTEM\_PAL  
     hardware.h, [222](#)  
  
 TAC\_REG  
     hardware.h, [213](#)  
 TACF\_16KHZ  
     hardware.h, [200](#)  
 TACF\_262KHZ  
     hardware.h, [201](#)  
 TACF\_4KHZ  
     hardware.h, [200](#)  
 TACF\_65KHZ  
     hardware.h, [201](#)  
 TACF\_START  
     hardware.h, [200](#)  
 TACF\_STOP  
     hardware.h, [200](#)  
 tile  
     OAM\_item\_t, [94](#)  
 TIM\_IFLAG  
     gb.h, [146](#)  
     msx.h, [283](#)  
     sms.h, [350](#)  
 TIMA\_REG  
     hardware.h, [213](#)  
 time  
     time.h, [385](#)  
 time.h  
     clock, [385](#)  
     CLOCKS\_PER\_SEC, [384](#)  
     time, [385](#)  
     time\_t, [384](#)  
 time\_t  
     time.h, [384](#)  
  
 TMA\_REG  
     hardware.h, [213](#)  
 TO\_FAR\_PTR  
     far\_ptr.h, [268](#)  
 to\_far\_ptr  
     far\_ptr.h, [269](#)  
 tolower  
     ctype.h, [117](#)  
 toupper  
     ctype.h, [117](#)  
 TRUE  
     types.h, [115](#)  
 true  
     stdbool.h, [372](#)  
 typeof.h  
     TYPEOF\_ARRAY, [386](#)  
     TYPEOF\_BIT, [386](#)  
     TYPEOF\_BITFIELD, [386](#)  
     TYPEOF\_CHAR, [386](#)  
     TYPEOF\_CPOINTER, [386](#)  
     TYPEOF\_EEPPPOINTER, [387](#)  
     TYPEOF\_FIXED16X16, [386](#)  
     TYPEOF\_FLOAT, [386](#)  
     TYPEOF\_FPOINTER, [386](#)  
     TYPEOF\_FUNCTION, [386](#)  
     TYPEOF\_GPOINTER, [386](#)  
     TYPEOF\_INT, [386](#)  
     TYPEOF\_IPOINTER, [387](#)  
     TYPEOF\_LONG, [386](#)  
     TYPEOF\_POINTER, [386](#)  
     TYPEOF\_PPOINTER, [387](#)  
     TYPEOF\_SBIT, [386](#)  
     TYPEOF\_SFR, [386](#)  
     TYPEOF\_SHORT, [386](#)  
     TYPEOF\_STRUCT, [386](#)  
     TYPEOF\_VOID, [386](#)  
 TYPEOF\_ARRAY  
     typeof.h, [386](#)  
 TYPEOF\_BIT  
     typeof.h, [386](#)  
 TYPEOF\_BITFIELD  
     typeof.h, [386](#)  
 TYPEOF\_CHAR  
     typeof.h, [386](#)  
 TYPEOF\_CPOINTER  
     typeof.h, [386](#)  
 TYPEOF\_EEPPPOINTER  
     typeof.h, [387](#)  
 TYPEOF\_FIXED16X16  
     typeof.h, [386](#)  
 TYPEOF\_FLOAT  
     typeof.h, [386](#)  
 TYPEOF\_FPOINTER  
     typeof.h, [386](#)  
 TYPEOF\_FUNCTION  
     typeof.h, [386](#)  
 TYPEOF\_GPOINTER  
     typeof.h, [386](#)



TYPEOF\_INT  
  typeof.h, 386  
TYPEOF\_IPOINTER  
  typeof.h, 387  
TYPEOF\_LONG  
  typeof.h, 386  
TYPEOF\_POINTER  
  typeof.h, 386  
TYPEOF\_PPOINTER  
  typeof.h, 387  
TYPEOF\_SBIT  
  typeof.h, 386  
TYPEOF\_SFR  
  typeof.h, 386  
TYPEOF\_SHORT  
  typeof.h, 386  
TYPEOF\_STRUCT  
  typeof.h, 386  
TYPEOF\_VOID  
  typeof.h, 386  
types.h  
  \_\_SIZE\_T\_DEFINED, 110, 111, 114  
  AT, 113  
  BANKED, 113  
  BOOLEAN, 113  
  BYTE, 113  
  clock\_t, 111, 112, 115  
  CRITICAL, 113  
  DWORD, 113  
  FALSE, 115  
  fixed, 114  
  INT16, 110, 111, 114  
  INT32, 110, 111, 115  
  INT8, 110, 111, 114  
  INTERRUPT, 113  
  LWORD, 113  
  NAKED, 113  
  NONBANKED, 113  
  NORETURN, 113  
  NULL, 115  
  OLDCALL, 112  
  POINTER, 115  
  PRESERVES\_REGS, 112  
  SFR, 113  
  size\_t, 111, 112, 115  
  TRUE, 115  
  UBYTE, 113  
  UDWORD, 113  
  UINT16, 110, 111, 114  
  UINT32, 110, 111, 115  
  UINT8, 110, 111, 114  
  ULWORD, 113  
  UWORD, 113  
  WORD, 113  
  Z88DK\_CALLEE, 114  
  Z88DK\_FASTCALL, 114  
UBYTE  
  types.h, 113  
UCHAR\_MAX  
  limits.h, 276  
UDWORD  
  types.h, 113  
UINT16  
  types.h, 110, 111, 114  
UINT16\_C  
  stdint.h, 377  
UINT16\_MAX  
  stdint.h, 375  
uint16\_t  
  stdint.h, 377  
uint2bcd  
  bcd.h, 118, 121, 123  
UINT32  
  types.h, 110, 111, 115  
UINT32\_C  
  stdint.h, 377  
UINT32\_MAX  
  stdint.h, 375  
uint32\_t  
  stdint.h, 378  
UINT8  
  types.h, 110, 111, 114  
UINT8\_C  
  stdint.h, 377  
UINT8\_MAX  
  stdint.h, 375  
uint8\_t  
  stdint.h, 377  
UINT\_FAST16\_MAX  
  stdint.h, 376  
uint\_fast16\_t  
  stdint.h, 378  
UINT\_FAST32\_MAX  
  stdint.h, 376  
uint\_fast32\_t  
  stdint.h, 378  
UINT\_FAST8\_MAX  
  stdint.h, 376  
uint\_fast8\_t  
  stdint.h, 378  
UINT\_LEAST16\_MAX  
  stdint.h, 375  
uint\_least16\_t  
  stdint.h, 378  
UINT\_LEAST32\_MAX  
  stdint.h, 375  
uint\_least32\_t  
  stdint.h, 378  
UINT\_LEAST8\_MAX  
  stdint.h, 375  
uint\_least8\_t  
  stdint.h, 378  
UINT\_MAX  
  limits.h, 276  
UINT\_MIN  
  limits.h, 276

- UINTMAX\_C
  - stdint.h, [377](#)
- UINTMAX\_MAX
  - stdint.h, [376](#)
- uintmax\_t
  - stdint.h, [378](#)
- UINTPTR\_MAX
  - stdint.h, [376](#)
- uintptr\_t
  - stdint.h, [378](#)
- uitoa
  - stdlib.h, [382](#)
- ULONG\_MAX
  - limits.h, [276](#)
- ULONG\_MIN
  - limits.h, [276](#)
- ultoa
  - stdlib.h, [382](#)
- ULWORD
  - types.h, [113](#)
- UNSIGNED
  - drawing.h, [133](#)
- USE\_C\_MEMCPY
  - provides.h, [95](#), [96](#)
- USE\_C\_STRCMP
  - provides.h, [95](#), [96](#)
- USE\_C\_STRCPY
  - provides.h, [95](#), [96](#)
- USHRT\_MAX
  - limits.h, [276](#)
- USHRT\_MIN
  - limits.h, [276](#)
- UWORD
  - types.h, [113](#)
- va\_arg
  - stdarg.h, [97](#), [98](#)
- va\_end
  - stdarg.h, [97](#), [98](#)
- va\_list
  - stdarg.h, [97](#), [98](#)
- va\_start
  - stdarg.h, [96–98](#)
- VBK\_ATTRIBUTES
  - hardware.h, [207](#), [222](#), [239](#)
- VBK\_BANK\_0
  - hardware.h, [206](#)
- VBK\_BANK\_1
  - hardware.h, [207](#)
- VBK\_REG
  - hardware.h, [215](#)
  - msx.h, [280](#)
  - sms.h, [348](#)
- VBK\_TILES
  - hardware.h, [207](#), [222](#), [239](#)
- VBL\_IFLAG
  - gb.h, [146](#)
  - msx.h, [282](#)
  - sms.h, [350](#)
- VDP\_ATTR\_SHIFT
  - hardware.h, [223](#), [240](#)
- VDP\_R0
  - hardware.h, [219](#), [234](#)
- VDP\_R1
  - hardware.h, [219](#), [235](#)
- VDP\_R10
  - hardware.h, [222](#), [237](#)
- VDP\_R2
  - hardware.h, [220](#), [235](#)
- VDP\_R3
  - hardware.h, [221](#), [236](#)
- VDP\_R4
  - hardware.h, [221](#), [236](#)
- VDP\_R5
  - hardware.h, [221](#), [236](#)
- VDP\_R6
  - hardware.h, [221](#), [236](#)
- VDP\_R7
  - hardware.h, [221](#), [236](#)
- VDP\_R8
  - hardware.h, [221](#), [236](#)
- VDP\_R9
  - hardware.h, [222](#), [237](#)
- VDP\_RBORDER
  - hardware.h, [221](#), [236](#)
- VDP\_REG\_MASK
  - hardware.h, [219](#), [234](#)
- VDP\_RSCX
  - hardware.h, [221](#), [236](#)
- VDP\_RSCY
  - hardware.h, [222](#), [237](#)
- VDP\_SAT\_TERM
  - hardware.h, [222](#), [239](#)
- VECTOR\_JOYPAD
  - isr.h, [242](#)
- VECTOR\_SERIAL
  - isr.h, [242](#)
- VECTOR\_STAT
  - isr.h, [242](#)
- VECTOR\_TIMER
  - isr.h, [242](#)
- version.h
  - \_\_GBDK\_VERSION, [275](#)
- vmemcpy
  - gb.h, [184](#)
  - msx.h, [294](#)
  - sms.h, [363](#)
- vmemset
  - gb.h, [188](#)
  - nes.h, [339](#)
- vsync
  - gb.h, [162](#)
  - msx.h, [290](#)
  - nes.h, [319](#)
  - sms.h, [358](#)
- w
  - \_fixed, [90](#)

wait\_int\_handler  
  gb.h, [158](#)

wait\_vbl\_done  
  gb.h, [162](#)  
  msx.h, [290](#)  
  nes.h, [319](#)  
  sms.h, [358](#)

waitpad  
  gb.h, [160](#)  
  msx.h, [291](#)  
  nes.h, [317](#)  
  sms.h, [359](#)

waitpadup  
  gb.h, [160](#)  
  msx.h, [291](#)  
  nes.h, [317](#)  
  sms.h, [359](#)

WCHAR\_MAX  
  stdint.h, [377](#)

WCHAR\_MIN  
  stdint.h, [377](#)

wchar\_t  
  stddef.h, [373](#)

WHITE  
  drawing.h, [132](#)

WINT\_MAX  
  stdint.h, [377](#)

WINT\_MIN  
  stdint.h, [377](#)

WORD  
  types.h, [113](#)

WRITE\_VDP\_CMD  
  msx.h, [288](#)  
  sms.h, [356](#)

WRITE\_VDP\_DATA  
  msx.h, [288](#)  
  sms.h, [356](#)

wrtchr  
  drawing.h, [135](#)

WX\_REG  
  hardware.h, [215](#)

WY\_REG  
  hardware.h, [215](#)

x  
  OAM\_item\_t, [94](#)

XOR  
  drawing.h, [132](#)

y  
  OAM\_item\_t, [94](#)

Z88DK\_CALLEE  
  types.h, [114](#)

Z88DK\_FASTCALL  
  types.h, [114](#)