

## GBDK 2020 Docs

Generated by Doxygen 1.8.17

Wed Mar 24 2021 21:19:10

<b>1 General Documentation</b>	<b>1</b>
1.1 Introduction	1
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. Compile Example projects	2
2.2 2. Use a Template	3
2.3 3. If you use GBTD / GBMB, get the fixed version	3
2.4 4. Review Coding Guidelines	3
2.5 5. Hardware and Resources	3
2.6 6. Set up C Source debugging	3
2.7 7. Try a GBDK Tutorial	3
2.8 8. Read up!	4
2.9 9. Need help?	4
<b>3 Links and Third-Party Tools</b>	<b>4</b>
3.1 SDCC Compiler Suite User Manual	4
3.2 Getting Help	4
3.3 Game Boy Documentation	4
3.4 Tutorials	4
3.5 Example code	4
3.6 Graphics Tools	5
3.7 Music drivers and tools	5
3.8 Emulators	5
3.9 Debugging tools	5
<b>4 Using GBDK</b>	<b>6</b>
4.1 Interrupts	6
4.1.1 Available Interrupts	6
4.1.2 Adding your own interrupt handler	7
4.1.3 Returning from Interrupts and STAT mode	7
4.2 What GBDK does automatically and behind the scenes	7
4.2.1 OAM (VRAM Sprite Attribute Table)	7
4.2.2 Font tiles when using stdio.h	7
4.2.3 Default Interrupt Service Handlers (ISRs)	7
4.3 Copying Functions to RAM and HIRAM	7
4.4 Mixing C and Assembly	8
4.4.1 Inline ASM within C source files	8
4.4.2 In Separate ASM files	8
4.5 Known Issues and Limitations	9
4.5.1 SDCC	9

<b>5 Coding Guidelines</b>	<b>9</b>
5.1 Learning C / C fundamentals	9
5.1.1 General C tutorials	9
5.1.2 Embedded C introductions	9
5.1.3 Game Boy games in C	9
5.2 Understanding the hardware	9
5.3 Writing optimal C code for the Game Boy and SDCC	9
5.3.1 Tools	9
5.3.2 Variables	9
5.3.3 Code structure	10
5.3.4 GBDK API/Library	10
5.3.5 Toolchain	11
5.3.6 chars and vararg functions	11
5.4 When C isn't fast enough	12
5.4.1 Calling convention	12
5.4.2 Variables and registers	12
5.4.3 Segments	12
<b>6 ROM/RAM Banking and MBCs</b>	<b>12</b>
6.1 ROM/RAM Banking and MBCs (Memory Bank Controllers)	12
6.1.1 Unbanked cartridges	12
6.1.2 MBC Banked cartridges (Memory Bank Controllers)	13
6.2 Working with Banks	13
6.2.1 Setting the ROM bank for a Source file	13
6.2.2 Setting the RAM bank for a Source file	13
6.2.3 Setting the MBC and number of ROM & RAM banks available	13
6.2.4 Banked Functions	14
6.2.5 Const Data (Variables in ROM)	14
6.2.6 Variables in RAM	14
6.2.7 Far Pointers	14
6.2.8 Bank switching	14
6.2.9 Restoring the current bank (after calling functions which change it without restoring)	15
6.2.10 Currently active bank: <code>_current_bank</code>	15
6.3 Auto-Banking	15
6.4 Errors related to banking (overflow, multiple writes to same location)	16
6.5 Bank space usage	16
6.5.1 Other important notes	16
6.6 Banking example projects	16
<b>7 GBDK Toolchain</b>	<b>16</b>
7.1 Overview	16
7.2 Data Types	17
7.3 Changing Important Addresses	17

7.4 Compiling programs . . . . .	17
7.4.1 Makefiles . . . . .	18
7.5 Build Tools . . . . .	18
7.5.1 lcc . . . . .	18
7.5.2 sdcc . . . . .	19
7.5.3 sdasgb . . . . .	19
7.5.4 bankpack . . . . .	19
7.5.5 slddgb . . . . .	19
7.5.6 ihxcheck . . . . .	19
7.5.7 makebin . . . . .	19
7.6 GBDK Utilities . . . . .	20
7.6.1 GBCompress . . . . .	20
7.6.2 PNG to Metasprite . . . . .	20
<b>8 Example Programs . . . . .</b>	<b>20</b>
8.1 banks (various projects) . . . . .	20
8.2 comm . . . . .	20
8.3 crash . . . . .	20
8.4 colorbar . . . . .	20
8.5 dscan . . . . .	20
8.6 filltest . . . . .	20
8.7 fonts . . . . .	20
8.8 galaxy . . . . .	21
8.9 gb-dtmf . . . . .	21
8.10 gbdecompress . . . . .	21
8.11 irq . . . . .	21
8.12 large map . . . . .	21
8.13 metasprites . . . . .	21
8.14 lcd isr wobble . . . . .	21
8.15 paint . . . . .	21
8.16 rand . . . . .	21
8.17 ram_fn . . . . .	21
8.18 rpn . . . . .	21
8.19 samptest . . . . .	22
8.20 sgb (various) . . . . .	22
8.21 sound . . . . .	22
8.22 space . . . . .	22
8.23 templates . . . . .	22
<b>9 Frequently Asked Questions (FAQ) . . . . .</b>	<b>22</b>
9.1 Frequently Asked Questions . . . . .	22
<b>10 Migrating to new GBDK Versions . . . . .</b>	<b>24</b>

10.1 GBDK 2020 versions	24
10.1.1 Porting to GBDK 2020 4.0.3	24
10.1.2 Porting to GBDK 2020 4.0.2	24
10.1.3 Porting to GBDK 2020 4.0.1	24
10.1.4 Porting to GBDK 2020 4.0	24
10.1.5 Porting to GBDK 2020 3.2	24
10.1.6 Porting to GBDK 2020 3.1.1	24
10.1.7 Porting to GBDK 2020 3.1	24
10.1.8 Porting to GBDK 2020 3.0.1	25
10.2 Historical GBDK versions	25
10.2.1 GBDK 1.1 to GBDK 2.0	25
<b>11 GBDK Releases</b>	<b>25</b>
11.1 GBDK 2020 Release Notes	25
11.1.1 GBDK 2020 4.0.3	25
11.1.2 GBDK 2020 4.0.2	26
11.1.3 GBDK 2020 4.0.1	26
11.1.4 GBDK 2020 4.0	27
11.1.5 GBDK 2020 3.2	28
11.1.6 GBDK 2020 3.1.1	28
11.1.7 GBDK 2020 3.1	28
11.1.8 GBDK 2020 3.0.1	28
11.1.9 GBDK 2020 3.0	28
11.2 Historical GBDK Release Notes	29
11.2.1 GBDK 2.96	29
11.2.2 GBDK 2.95-3	29
11.2.3 GBDK 2.95-2	29
11.2.4 GBDK 2.95	29
11.2.5 GBDK 2.94	30
11.2.6 GBDK 2.93	30
11.2.7 GBDK 2.92-2 for win32	31
11.2.8 GBDK 2.92	31
11.2.9 GBDK 2.91	32
11.2.10 GBDK 2.1.5	32
<b>12 Toolchain settings</b>	<b>32</b>
12.1 lcc settings	32
12.2 sdcc settings	33
12.3 sdasgb settings	36
12.4 bankpack settings	37
12.5 sldlgb settings	37
12.6 ihxcheck settings	37
12.7 makebin settings	37

12.8 gbcompress settings . . . . .	38
12.9 png2mtspr settings . . . . .	38
<b>13 Todo List</b>	<b>38</b>
<b>14 Module Index</b>	<b>39</b>
14.1 C modules . . . . .	39
<b>15 Data Structure Index</b>	<b>39</b>
15.1 Data Structures . . . . .	39
<b>16 File Index</b>	<b>39</b>
16.1 File List . . . . .	39
<b>17 Module Documentation</b>	<b>40</b>
17.1 List of gbdk fonts . . . . .	40
17.1.1 Description . . . . .	40
17.1.2 Variable Documentation . . . . .	41
<b>18 Data Structure Documentation</b>	<b>41</b>
18.1 __far_ptr Union Reference . . . . .	41
18.1.1 Detailed Description . . . . .	41
18.1.2 Field Documentation . . . . .	41
18.2 _fixed Union Reference . . . . .	42
18.2.1 Detailed Description . . . . .	42
18.2.2 Field Documentation . . . . .	42
18.3 atomic_flag Struct Reference . . . . .	42
18.3.1 Field Documentation . . . . .	42
18.4 joypads_t Struct Reference . . . . .	43
18.4.1 Detailed Description . . . . .	43
18.4.2 Field Documentation . . . . .	43
18.5 metasprite_t Struct Reference . . . . .	43
18.5.1 Detailed Description . . . . .	44
18.5.2 Field Documentation . . . . .	44
18.6 OAM_item_t Struct Reference . . . . .	44
18.6.1 Detailed Description . . . . .	44
18.6.2 Field Documentation . . . . .	45
18.7 sfont_handle Struct Reference . . . . .	45
18.7.1 Detailed Description . . . . .	45
18.7.2 Field Documentation . . . . .	45
18.8 smalloc_hunk Struct Reference . . . . .	45
18.8.1 Field Documentation . . . . .	46
<b>19 File Documentation</b>	<b>47</b>
19.1 /home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/01_getting_started.md File Reference . . . . .	47

19.2	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/02_links_and_tools.md File Reference	47
19.3	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/03_using_gbdk.md File Reference	47
19.4	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/04_coding_guidelines.md File Reference	47
19.5	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/05_banking_mbcx.md File Reference	47
19.6	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/06_toolchain.md File Reference	47
19.7	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/07_sample_programs.md File Reference	47
19.8	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/08_faq.md File Reference	47
19.9	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/09_migrating_new_versions.md File Reference	47
19.10	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/10_release_notes.md File Reference	47
19.11	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/20_toolchain_settings.md File Reference	47
19.12	/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/docs_index.md File Reference	47
19.13	asm/gbz80/provides.h File Reference	47
19.13.1	Macro Definition Documentation	47
19.14	asm/gbz80/stdarg.h File Reference	48
19.14.1	Macro Definition Documentation	48
19.14.2	Typedef Documentation	48
19.15	stdarg.h File Reference	48
19.16	asm/gbz80/types.h File Reference	48
19.16.1	Detailed Description	49
19.16.2	Macro Definition Documentation	49
19.16.3	Typedef Documentation	49
19.17	asm/types.h File Reference	50
19.17.1	Detailed Description	50
19.17.2	Typedef Documentation	50
19.18	types.h File Reference	51
19.18.1	Detailed Description	51
19.18.2	Macro Definition Documentation	51
19.18.3	Typedef Documentation	51
19.19	assert.h File Reference	51
19.19.1	Macro Definition Documentation	52
19.19.2	Function Documentation	52
19.20	bcd.h File Reference	52
19.20.1	Detailed Description	52
19.20.2	Macro Definition Documentation	52
19.20.3	Typedef Documentation	53
19.20.4	Function Documentation	53
19.21	ctype.h File Reference	54
19.21.1	Detailed Description	54
19.21.2	Function Documentation	54
19.22	gb/bgb_emu.h File Reference	55
19.22.1	Detailed Description	56

19.22.2 Macro Definition Documentation . . . . .	56
19.22.3 Function Documentation . . . . .	57
19.23 gb/cgb.h File Reference . . . . .	57
19.23.1 Detailed Description . . . . .	58
19.23.2 Macro Definition Documentation . . . . .	58
19.23.3 Function Documentation . . . . .	60
19.24 gb/console.h File Reference . . . . .	62
19.24.1 Detailed Description . . . . .	62
19.24.2 Function Documentation . . . . .	62
19.25 gb/crash_handler.h File Reference . . . . .	63
19.25.1 Detailed Description . . . . .	63
19.25.2 Function Documentation . . . . .	63
19.26 gb/drawing.h File Reference . . . . .	63
19.26.1 Detailed Description . . . . .	64
19.26.2 Macro Definition Documentation . . . . .	64
19.26.3 Function Documentation . . . . .	65
19.27 gb/far_ptr.h File Reference . . . . .	68
19.27.1 Detailed Description . . . . .	68
19.27.2 Macro Definition Documentation . . . . .	69
19.27.3 Typedef Documentation . . . . .	71
19.27.4 Function Documentation . . . . .	71
19.27.5 Variable Documentation . . . . .	71
19.28 gb/font.h File Reference . . . . .	72
19.28.1 Detailed Description . . . . .	72
19.28.2 Macro Definition Documentation . . . . .	72
19.28.3 Typedef Documentation . . . . .	72
19.28.4 Function Documentation . . . . .	73
19.29 gb/gb.h File Reference . . . . .	73
19.29.1 Detailed Description . . . . .	77
19.29.2 Macro Definition Documentation . . . . .	77
19.29.3 Typedef Documentation . . . . .	83
19.29.4 Function Documentation . . . . .	83
19.29.5 Variable Documentation . . . . .	105
19.30 gb/gbdecompress.h File Reference . . . . .	106
19.30.1 Detailed Description . . . . .	106
19.30.2 Function Documentation . . . . .	106
19.30.3 Variable Documentation . . . . .	107
19.31 gb/hardware.h File Reference . . . . .	108
19.31.1 Detailed Description . . . . .	109
19.31.2 Macro Definition Documentation . . . . .	109
19.31.3 Variable Documentation . . . . .	109
19.32 gb/malloc.h File Reference . . . . .	112



19.32.1 Detailed Description . . . . .	113
19.32.2 Macro Definition Documentation . . . . .	113
19.32.3 Typedef Documentation . . . . .	113
19.32.4 Function Documentation . . . . .	113
19.32.5 Variable Documentation . . . . .	113
19.33 gb/metasprites.h File Reference . . . . .	113
19.33.1 Detailed Description . . . . .	114
19.33.2 Macro Definition Documentation . . . . .	114
19.33.3 Typedef Documentation . . . . .	114
19.33.4 Function Documentation . . . . .	115
19.33.5 Variable Documentation . . . . .	117
19.34 gb/sample.h File Reference . . . . .	117
19.34.1 Detailed Description . . . . .	118
19.34.2 Function Documentation . . . . .	118
19.35 gb/sgb.h File Reference . . . . .	118
19.35.1 Detailed Description . . . . .	118
19.35.2 Macro Definition Documentation . . . . .	119
19.35.3 Function Documentation . . . . .	120
19.36 gbdk-lib.h File Reference . . . . .	121
19.36.1 Detailed Description . . . . .	121
19.37 limits.h File Reference . . . . .	121
19.37.1 Macro Definition Documentation . . . . .	121
19.38 rand.h File Reference . . . . .	122
19.38.1 Detailed Description . . . . .	122
19.38.2 Function Documentation . . . . .	123
19.39 setjmp.h File Reference . . . . .	123
19.39.1 Macro Definition Documentation . . . . .	124
19.39.2 Typedef Documentation . . . . .	124
19.39.3 Function Documentation . . . . .	124
19.40 stdatomic.h File Reference . . . . .	125
19.40.1 Function Documentation . . . . .	125
19.41 stdbool.h File Reference . . . . .	125
19.41.1 Macro Definition Documentation . . . . .	125
19.42 stddef.h File Reference . . . . .	125
19.42.1 Macro Definition Documentation . . . . .	126
19.42.2 Typedef Documentation . . . . .	126
19.43 stdint.h File Reference . . . . .	126
19.43.1 Macro Definition Documentation . . . . .	128
19.43.2 Typedef Documentation . . . . .	131
19.44 stdio.h File Reference . . . . .	132
19.44.1 Detailed Description . . . . .	132
19.44.2 Function Documentation . . . . .	132

19.45 stdlib.h File Reference . . . . .	133
19.45.1 Macro Definition Documentation . . . . .	134
19.45.2 Function Documentation . . . . .	134
19.46 stdnoreturn.h File Reference . . . . .	137
19.46.1 Macro Definition Documentation . . . . .	137
19.47 string.h File Reference . . . . .	137
19.47.1 Detailed Description . . . . .	137
19.47.2 Function Documentation . . . . .	138
19.47.3 Variable Documentation . . . . .	140
19.48 time.h File Reference . . . . .	141
19.48.1 Detailed Description . . . . .	141
19.48.2 Macro Definition Documentation . . . . .	141
19.48.3 Typedef Documentation . . . . .	141
19.48.4 Function Documentation . . . . .	141
19.49 typedef.h File Reference . . . . .	142
19.49.1 Macro Definition Documentation . . . . .	142
<b>Index</b>	<b>145</b>

## 1 General Documentation

- [Getting Started](#)
- [Links and Third-Party Tools](#)
- [Using GBDK](#)
- [Coding Guidelines](#)
- [ROM/RAM Banking and MBCs](#)
- [GBDK Toolchain](#)
- [Example Programs](#)
- [Frequently Asked Questions \(FAQ\)](#)
- [Migrating to new GBDK Versions](#)
- [GBDK Releases](#)
- [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.

## 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/Za10/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 system 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 with 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

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

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. Compile Example projects

Make sure your GBDK-2020 installation is working correctly by compiling some of the included [example projects](#). 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`.

If everything works and there are no errors reported each example sub-folder should have it's on .gb ROM file.

## 2.2 2. 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 `make.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.

## 2.3 3. 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](#).

## 2.4 4. 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](#).

## 2.5 5. Hardware and Resources

If you have a specific project in mind, consider what hardware 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 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](#)

## 2.6 6. 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.

## 2.7 7. 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 principals 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).

## 2.8 8. 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.

## 2.9 9. Need help?

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

## 3 Links and Third-Party Tools

This is a brief list of useful tools and information. It is not meant to be complete or exhaustive, for a larger list see [awesome\\_gb](#)

### 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>

### 3.2 Getting Help

- GBDK Discord community:  
<https://github.com/Zal0/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 List**  
A list of Game Boy/Color development resources, tools, docs, related projects and homebrew.  
<https://gbdev.io/list.html>

### 3.4 Tutorials

- **Gaming Monsters Tutorials**  
Several video tutorials and code for making games with GBDK/GBDK-2020.  
<https://www.youtube.com/playlist?list=PLeEj4c2zF7PaFv5MPYhNAkBGkx4iPGJo>  
<https://github.com/gingemonster/GamingMonstersGameBoySampleCode>

### 3.5 Example code

- **Simplified GBDK examples**  
[https://github.com/mrombout/gbdk\\_playground/commits/master](https://github.com/mrombout/gbdk_playground/commits/master)

## 3.6 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.
  - **Use this updated version:** (has const export fixed and other improvements):  
[https://github.com/untoxa/GBTD\\_GBMB](https://github.com/untoxa/GBTD_GBMB)
  - This older version is **not recommended**:  
<http://www.devr.com/gb/hmgd/intro.html> (old, original tools)
  - A GIMP plugin for import/export:  
<https://github.com/bbbbr/gimp-tilemap-gb> (GIMP plugin to read/write GBR/GBM files)
- **Tilemap Studio**  
A tilemap editor for Game Boy, GBC, GBA, or SNES projects.  
<https://github.com/Rangi42/tilemap-studio/>

## 3.7 Music drivers and tools

- **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/>
- **hUGEdriver**  
A tracker and music driver that works with GBDK and RGBDS. It is smaller, more efficient and more versatile than gbt\_player.  
<https://github.com/untoxa/hUGEBuild>  
<https://github.com/SuperDisk/hUGEDriver>  
<https://github.com/SuperDisk/hUGETracker>

## 3.8 Emulators

- **BGB**  
Accurate emulator, has useful debugging tools.  
<http://bgb.bircd.org/>
- **Emulicious**  
An accurate emulator with extensive tools including source level debugging.  
<https://emulicious.net/>

## 3.9 Debugging tools

- **Emulicious debug adapter**  
Provides source-level debugging in VS Code that works with GBDK2020.  
<https://marketplace.visualstudio.com/items?itemName=emulicious.emulicious-debugger>
- **romusage**  
Calculate used and free space in banks (ROM/RAM) and warn about errors such as bank overflows.  
<https://github.com/bbbbr/romusage>
- **noi2sym.py**  
Convert .noi files into a symbol format compatible with BGB. Allows BGB to recognize variables and functions based on address.  
<https://github.com/untoxa/hUGEBuild/blob/master/tools/noi2sym.py>

- **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>

## 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(UBYTE ier)` and the interrupt service routine (ISR) linkers `add_VBL()`, `add_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()` and `remove_TIM()`
  - Example project: `tim`
- SIO : Serial Link I/O transfer end
  - The default SIO ISR gets installed automatically if any of the standard SIO calls are used. These calls include `add_SIO()`, `remove_SIO()`, `send_byte()`, `receive_byte()`.
  - The default SIO ISR cannot be removed once installed. Only secondary chained SIO ISRs (added with `add_SIO()`) can be removed.
  - See `add_SIO()` and `remove_SIO()`
  - Example project: `comm`
- JOY : Transition from high to low of a joypad button
  - See `add_JOY()` and `remove_JOY()`

### 4.1.2 Adding your own interrupt handler

It is possible to install your own interrupt handlers (in C or in assembly) for any of these interrupts. Up to 4 chained handlers may be added, with the last added being called last. If the [remove\\_VBL\(\)](#) function is to be called, only three may be added for VBL.

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

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

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

### 4.1.3 Returning from Interrupts and STAT mode

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

## 4.2 What GBDK does automatically and behind the scenes

### 4.2.1 OAM (VRAM Sprite Attribute Table)

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

### 4.2.2 Font tiles when using `stdio.h`

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

### 4.2.3 Default Interrupt Service Handlers (ISRs)

- V-Blank: A default V-Blank ISR is installed on startup which copies the Shadow OAM to the hardware OAM and increments the global [sys\\_time](#) variable once per frame.
- Serial Link I/O: If any of the GBDK serial link functions are used such as [send\\_byte\(\)](#) and [receive\\_byte\(\)](#), the default SIO serial link handler will be installed automatically at compile-time.

## 4.3 Copying Functions to RAM and HIRAM

The `ram_function` example project included with GBDK demonstrates copying functions to RAM and HIRAM. It is possible to copy functions to RAM and HIRAM (using the [memcpy\(\)](#) and [hramcpy\(\)](#) functions), and execute them from C. The compiler automatically generates two symbols 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. Ensure you have enough free space in RAM or HIRAM for copying a function.

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

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

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



## 4.4 Mixing C and Assembly

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

### 4.4.1 Inline ASM within C source files

Example:

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

Another Example:

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

### 4.4.2 In Separate ASM files

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

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

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

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

`main.c`

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

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

`add.s`

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

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

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

## 4.5 Known Issues and Limitations

### 4.5.1 SDCC

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

## 5 Coding Guidelines

### 5.1 Learning C / C fundamentals

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

#### 5.1.1 General C tutorials

- <https://www.learn-c.org/>
- <https://www.tutorialspoint.com/cprogramming/index.htm>

#### 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/list.html#c>

### 5.2 Understanding the hardware

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

### 5.3 Writing optimal C code for the Game Boy and SDCC

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

#### 5.3.1 Tools

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

\_\_Use of [toxa's updated GBTD/GBMB](#) is highly recommended.\_\_

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

#### 5.3.2 Variables

- Use 8-bit values as much as possible. They will be much more efficient and compact than 16 and 32 bit types.
- Prefer unsigned variables to signed ones: The code generated will be generally more efficient, especially when comparing two values.

- Use explicit types so you always know the size of your variables. `INT8`, `UINT8`, `INT16`, `UINT16`, `INT32`, `UINT32` or `BYTE`, `UBYTE`, `WORD`, `UWORD`, `LWORD`, `ULWORD`. Types are defined in [asm/types.h](#) and [asm/gbz80/types.h](#)
- Global and local static variables are generally more efficient than local non-static variables (which go on the stack and are slower and can result in slower code).
- `const` keyword: Use `const` for arrays, structs and variables with read-only (constant) data. It will reduce ROM, RAM and CPU usage significantly. Non-`const` values are loaded from ROM into RAM inefficiently, and there is no benefit in loading them into the limited available RAM if they aren't going to be changed.
- For calculated values that don't change, pre-compute results once and store the result. Using lookup-tables and the like can improve speed and reduce code size. Macros can sometimes help. It may be beneficial to do the calculations with an outside tool and then include the result as C code in a `const` array.
- Use an advancing pointer (`someStruct->var = x; someStruct++`) to loop through arrays of structs instead of using indexing each time in the loop `someStruct[i].var = x`.
- When modifying variables that are also changed in an Interrupt Service Routine (ISR), wrap them the relevant code block in a `__critical { }` block. See <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.3.9>
- When using constants and literals the `U`, `L` and `UL` postfixes can be used.
  - `U` specifies that the constant is unsigned
  - `L` specifies that the constant is long.
  - NOTE: In SDCC 3.6.0, the default for `char` changed from signed to unsigned. The manual says to use `--fsigned-char` for the old behavior, this option flag is included by default when compiling through [lcc](#).

### 5.3.3 Code structure

- When processing for a given frame is done and it is time to wait before starting the next frame, [wait\\_vbl\\_done\(\)](#) can be used. It uses `HALT` to put the CPU into a low power state until processing resumes. The CPU will wake up and resume processing at the end of the current frame when the Vertical Blanking interrupt is triggered.
- Minimize use of multiplication, modulo with non-powers of 2, and division with non-powers of 2. These operations have no corresponding CPU instructions (software functions), and hence are time costly.
  - SDCC has some optimizations for:
    - \* Division by powers of 2. For example `n /= 4u` will be optimized to `n >>= 2`.
    - \* Modulo by powers of 2. For example: `(n % 8)` will be optimized to `(n & 0x7)`.
  - If you need decimal numbers to count or display a score, you can use the GBDK BCD ( [binary coded decimal](#)) number functions. See: [bcd.h](#) and the BCD example project included with GBDK.
- Avoid long lists of function parameters. Passing many parameters can add overhead, especially if the function is called often. When applicable globals and local static vars can be used instead.
- Use inline functions if the function is short. (with the `inline` keyword, such as `inline UINT8 myFunction() { ... }`)
- Do not use recursive functions

### 5.3.4 GBDK API/Library

- `stdio.h`: If you have other ways of printing text, avoid including [stdio.h](#) and using functions such as [printf\(\)](#). Including it will use a large number of the background tiles for font characters. If `stdio.h` is not included then that space will be available for use with other tiles instead.
- `drawing.h`: The Game Boy graphics hardware is not well suited to frame-buffer style graphics such as the kind provided in [drawing.h](#). Due to that, most drawing functions (rectangles, circles, etc) will be slow. When possible it's much faster and more efficient to work with the tiles and tile maps that the Game Boy hardware is built around.

- `waitpad()` and `waitpadup` check for input in a loop that doesn't HALT at all, so the CPU will be maxed out until it returns. One alternative is to write a function with a loop that checks input with `joypad()` and then waits a frame using `wait_vbl_done()` (which idles the CPU while waiting) before checking input again.

### 5.3.5 Toolchain

- See SDCC optimizations: <http://sdcc.sourceforge.net/doc/sdccman.pdf#section.8.1>
- Use profiling. Look at the ASM generated by the compiler, write several versions of a function, compare them and choose the faster one.
- Use the SDCC `--max-allocs-per-node` flag with large values, such as 50000. `--opt-code-speed` has a much smaller effect.
  - GBDK-2020 (after v4.0.1) compiles the library with `--max-allocs-per-node 50000`, but it must be turned on for your own code.  
(example: `lcc ... -Wf--max-allocs-per-mode 50000` or `sdcc ... --max-allocs-per-mode 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 Releases](#)
- Learn some ASM and inspect the compiler output to understand what the compiler is doing and how your code gets translated. This can help with writing better C code and with debugging.

### 5.3.6 chars and vararg functions

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:

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

Also See:

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

## 5.4 When C isn't fast enough

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

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

### 5.4.1 Calling convention

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

The parameters to a function are pushed in right to left order with no aligning - so a byte takes up a byte on the stack instead of the more natural word. So for example the function `int store_byte( UWORD addr, UBYTE byte)` would push 'byte' onto the stack first then addr using a total of three bytes. As the return address is also pushed, the stack would contain:

```
At SP+0 - the return address
At SP+2 - addr
At SP+4 - byte
```

Note that the arguments that are pushed first are highest in the stack due to how the Game Boy's stack grows downwards.

The function returns in DE.

### 5.4.2 Variables and registers

C normally expects registers to be preserved across a function call. However in the case above as DE is used as the return value and HL is used for anything, only BC needs to be preserved.

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

The use of segments for code, data and variables is more noticeable in assembler. GBDK and SDCC define a number of default segments - `_CODE`, `_DATA` and `_BSS`. Two extra segments `_HEADER` and `_HEAP` exist for the Game Boy header and malloc heap respectively.

The order these segments are linked together is determined by `crt0.s` and is currently `_CODE` in ROM, then `_DATA`, `_BSS`, `_HEAP` in WRAM, with `STACK` at the top of WRAM. `_HEAP` is placed after `_BSS` so that all spare memory is available for the malloc routines. To place code in other than the first two banks, use the segments `_CODE_x` where x is the 16kB bank number.

As the `_BSS` segment occurs outside the ROM area you can only use `.ds` to reserve space in it.

While you don't have to use the `_CODE` and `_DATA` distinctions in assembler you may wish to do so consistency.

## 6 ROM/RAM Banking and MBCs

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

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

#### 6.1.1 Unbanked cartridges

Cartridges with no MBC controller are unbanked, 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.

- Bank 0 of the ROM is located in the region at 0000h – 3FFFh. It is *usually* fixed (unbanked) and cannot be switched out for another bank.
- The higher region at 4000h to 7FFFh is used for switching between different ROM banks.

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

## 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 `UNBANKED` keyword to define a function as unbanked 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 RAM bank `-Wf-ba<N>`. Example (RAM bank 3): `-Wf-bo3`

### 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](#).

- `-Wl-yo<N>` where `<N>` is the number of ROM banks. 2, 4, 8, 16, 32, 64, 128, 256, 512
- `-Wl-ya<N>` where `<N>` is the number of RAM banks. 2, 4, 8, 16, 32
- `-Wl-yt<N>` where `<N>` is the type of MBC cartridge (see below).

The following MBC settings are available when using the `makebin` MBC switch.

```
# From Makebin source:
#
# -Wl-yt<NN> where <NN> is one of the numbers below
#
# 0147: Cartridge type:
# 0-ROM ONLY          12-ROM+MBC3+RAM
# 1-ROM+MBC1          13-ROM+MBC3+RAM+BATT
# 2-ROM+MBC1+RAM      19-ROM+MBC5
# 3-ROM+MBC1+RAM+BATT 1A-ROM+MBC5+RAM
# 5-ROM+MBC2          1B-ROM+MBC5+RAM+BATT
# 6-ROM+MBC2+BATTERY  1C-ROM+MBC5+RUMBLE
```

```
# 8-ROM+RAM          1D-ROM+MBC5+RUMBLE+SRAM
# 9-ROM+RAM+BATTERY  1E-ROM+MBC5+RUMBLE+SRAM+BATT
# B-ROM+MMM01        1F-Pocket Camera
# C-ROM+MMM01+SRAM    FD-Bandai TAMA5
# D-ROM+MMM01+SRAM+BATT FE - Hudson HuC-3
# F-ROM+MBC3+TIMER+BATT FF - Hudson HuC-1
# 10-ROM+MBC3+TIMER+RAM+BATT
# 11-ROM+MBC3
```

## 6.2.4 Banked Functions

Banked functions can be called as follows.

- When defined with the `BANKED` keyword. Example: `void my_function() BANKED { do stuff }` in a source file which has had its bank set (see above).
- Using [far\\_pointers](#)
- When defined with an area set up using the `__addressmod` keyword (See the `banks_new` example project and the SDCC manual for details)
- Using [SWITCH\\_ROM\\_MBC1\(\)](#) (and related functions for other MBCs) to manually switch in the required bank and then call the function.

Unbanked functions (either in fixed Bank 0, or in an Unbanked ROM with no MBC)

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

**Todo** Fill in this info for Banked Functions Banked functions (located in a switchable ROM bank)

- May call functions in any bank: ?
- May use data in any bank: **NO** (may only use data from currently active banks)

Limitations:

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

## 6.2.5 Const Data (Variables in ROM)

**Todo** Const Data (Variables in ROM)

## 6.2.6 Variables in RAM

**Todo** Variables in RAM

## 6.2.7 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.

## 6.2.8 Bank switching

You can manually switch banks using the [SWITCH\\_ROM\\_MBC1\(\)](#), [SWITCH\\_RAM\\_MBC1\(\)](#), and other related macros. See `banks.c` project for an example.

Note: You can only do a `switch_rom_bank` call from unbanked `_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 unbanked `_CODE`.

### 6.2.9 Restoring the current bank (after calling functions which change it without restoring)

If a function call is made (for example inside an ISR) which changes the bank *without* restoring it, then the `_current_bank` variable should be saved and then restored.

For example, **instead** of this code:

```
void vbl_music_isr(void)
{
    // A function which changes the bank and
    // *doesn't* restore it after changing.
    some_function();
}
```

It should be:

```
void vbl_music_isr(void)
{
    // Save the current bank
    UBYTE _saved_bank = _current_bank;
    // A function which changes the bank and
    // *doesn't* restore it after changing.
    some_function();
    // Now restore the current bank
    SWITCH_ROM_MBC5(_saved_bank);
}
```

### 6.2.10 Currently active bank: `_current_bank`

The global variable `_current_bank` is updated automatically when calling `SWITCH_ROM_MBC1()` and `SWITCH_ROM_MBC5`, or when a BANKED function is called.

## 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 bank for the source file to 255: `#pragma bank 255`
- Create a constant with no value to store the bank number for the source file: `const void __at(255) __bank_<name-for-a-given-source-file>;`.  
This constant can then be used for obtaining that file's bank number with `(UINT8)&__bank_<name-for-a-given-source-file>`.

Example: `level_1_map.c`

```
#pragma bank 255
const void __at(255) __bank_level_1_map;

const UINT8 my_level_1_map[] = {... some map data here ...};
```

Accessing that data: `main.c`

```
SWITCH_ROM_MBC1( (UINT8)&__bank_level_1_map );
// Do something with my_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 it's previously assigned bank until a source change causes the compiler to rebuild it to an object file again which resets it's 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

Limitations:

- At this time, the constant entries that get rewritten with the assigned bank (`const void at(255) __bank_↔ <name-you-want-to-use-for-that-source-file>;`) **cannot** be used from the source file they are declared in. In that case SDCC converts the bank number before [bankpack](#) has a chance to rewrite it. It may be referenced from any other source file, but not it's own.

## 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 third-party [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 third-party [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_MBC1` - 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 it's 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 of in GBDK 2020 4.0.2 or later, instead of having to manually specify which bank a source file will reside it.

# 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 [sdlrgb](#)
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/gbz80/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 [sdlrgb](#) 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 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

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>

## 7.5 Build Tools

### 7.5.1 lcc

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

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

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

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

### 7.5.2 sdcc

SDCC C Source compiler

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

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

### 7.5.3 sdasgb

SDCC Assembler for the gameboy

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>`

Limitations

- `__banked` functions cannot be called from within the same source file they are declared in.
- With data it is easier, because if you access data from the code in the same bank you don't need to switch the bank (access to `__bank_*` symbol).

### 7.5.5 sdldgb

The SDCC linker for the gameboy.

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

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

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

### 7.5.6 ihxcheck

IHX file validator

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

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

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

### 7.5.7 makebin

IHX to ROM converter

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

Converts .ihx files produced by [sdldgb](#) into ROM files (.gb, .gbc).

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

## 7.6 GBDK Utilities

### 7.6.1 GBCompress

Compression utility

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

Compresses (and decompresses) binary file data with the gbcompress algorithm (also used in GBTD/GBMB).

Decompression support is available in GBDK, see [gb\\_decompress\(\)](#).

### 7.6.2 PNG to Metasprite

Tool for converting PNGs into GBDK format MetaSprites

For detailed settings see [png2mtspr-settings](#)

**Todo** Document png2mtspr

## 8 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.

### 8.1 banks (various projects)

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

### 8.2 comm

Illustrates how to use communication routines.

### 8.3 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.

### 8.4 colorbar

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

### 8.5 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
```

### 8.6 filltest

Demonstrates various graphics routines.

### 8.7 fonts

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

## 8.8 galaxy

A C translation of the space.s assembly program.

## 8.9 gb-dtmf

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

## 8.10 gbdecompress

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

## 8.11 irq

Illustrates how to install interrupt handlers.

## 8.12 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.

## 8.13 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

## 8.14 lcd isr wobble

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

## 8.15 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
```

## 8.16 rand

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

## 8.17 ram\_fn

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

## 8.18 rpnc

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

## 8.19 samptest

Demonstration of playing a sound sample.

## 8.20 sgb (various)

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

## 8.21 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
```

## 8.22 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
```

## 8.23 templates

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

# 9 Frequently Asked Questions (FAQ)

## 9.1 Frequently Asked Questions

- 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?
  - See [setting\\_mbc\\_and\\_rom\\_ram\\_banks](#)

- What do these kinds of warnings / errors mean? `WARNING: possibly wrote twice at addr 4000 (93->3E) Warning: Write from one bank spans into the next. 7ff7 -> 8016 (bank 1 -> 2)`

You may have a overflow in one of your ROM banks. If there is more data allocated to a bank than it can hold it then will spill over into the next bank. The warnings are generated by [ihxcheck](#) during conversion of an .ihx file into a ROM file.

See the section [ROM/RAM Banking and MBCs](#) for more details about how banks work and what their size is. You may want to use a tool such as [romusage](#) to calculate the amount of free and used space.

- 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](#)
- What flags should be enabled for debugging?
  - You can use the [lcc debug flag](#)
- 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](#)



## 10 Migrating to new GBDK Versions

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

### 10.1 GBDK 2020 versions

#### 10.1.1 Porting to GBDK 2020 4.0.3

- No significant changes required

#### 10.1.2 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.

#### 10.1.3 Porting to GBDK 2020 4.0.1

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

#### 10.1.4 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`.
- OAM symbol has been renamed to `_shadow_OAM`, that allows accessing shadow OAM directly from C code

#### 10.1.5 Porting to GBDK 2020 3.2

- No significant changes required

#### 10.1.6 Porting to GBDK 2020 3.1.1

- No significant changes required

#### 10.1.7 Porting to GBDK 2020 3.1

- No significant changes required

### 10.1.8 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 usable symbols turn on `.noi` output (LCC argument: `-Wl-j`) can be enabled and you can use `noi2sym`
  - ?? 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

## 10.2 Historical GBDK versions

### 10.2.1 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.

## 11 GBDK Releases

The GBDK 2020 releases can be found on Github: <https://github.com/Zal0/gbdk-2020/releases>

### 11.1 GBDK 2020 Release Notes

#### 11.1.1 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

### 11.1.2 GBDK 2020 4.0.2

2021/01/17

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

### 11.1.3 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

#### 11.1.4 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, slddbg 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

### 11.1.5 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

### 11.1.6 GBDK 2020 3.1.1

2020/05/17

- Fixed issues with `libgcc_s_dw2-1.dll`

### 11.1.7 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
- `USE_SFR_FOR_REG` is the default now 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/Zal0/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>

### 11.1.8 GBDK 2020 3.0.1

2020/04/12

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

### 11.1.9 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

## 11.2 Historical GBDK Release Notes

### 11.2.1 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.

### 11.2.2 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.

### 11.2.3 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.

### 11.2.4 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

### 11.2.5 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_RAM` EG=1
  - Added `remove_VBL` etc functions.
  - Documented the libs - see the gbdk-doc tarball distributed seperatly.
  - Two dimensional arrays seem to be broken.

### 11.2.6 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 don't 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.

### 11.2.7 GBDK 2.92-2 for win32

26th March, 2000

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

- win32 only. Takes care of some of the install bugs, including:
  - Now auto detects where it is installed. This can be overridden using set GBDKDIR=...
  - Problems with the installer (now uses WinZip)
  - Problems with the temp directory Now scans TMP, TEMP, TMPDIR and finally c: tmp
  - cygwin1.dll and 'make' are no longer required gbdk is now built using mingw32 which is win32 native make.bat is automatically generated from the Makefile
  - I've reverted to using WORD for signed 16 bit etc. GBDK\_2\_COMPAT is no longer required.

WORDS are now back to signed. GBDK\_2\_COMPAT is no longer needed. Temporary files are created in TMP, TEMP, or TMPDIR instead of c: tmp The installer is no more as it's not needed. There is a WinZip wrapped version for those with the extra bandwidth :). gbdk autodetects where it is installed - no more environment variables. cygwin1.dll and make are no longer required - gbdk is now compiled with mingw32.

See the ChangeLog section in the README for more information.

21st March, 2000

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

18th March, 2000

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

### 11.2.8 GBDK 2.92

8th March, 2000

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

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



### 11.2.9 GBDK 2.91

27th Feb, 2000

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

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

23rd Feb, 2000

First release of gbdk/sdcc. This is an early release - the only binary is for Linux and the source is only available through cvs. If your interested in the source, have a look at the cvs repository gbdk-support first, which will download all the rest of the code. Alternatively, look at gbdk-support and gbdk-lib at cvs.gbdk.sourceforge.net and sdcc at cvs.sdcc.sourceforge.net. I will be working on binaries for Win32 and a source tar ball soon. Please report any bugs through the bugs link above.

31st Jan, 2000

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

8th Jan, 2000

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

6th Jan, 2000

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

18th Oct, 1999

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

### 11.2.10 GBDK 2.1.5

17th Oct, 1999

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

## 12 Toolchain settings

### 12.1 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 turn on --debug for compiler, -y (.cdb) and -j (.noi) for linker
-Dname -Dname=def define the preprocessor symbol 'name'
-E  run only the preprocessor on the named C programs and unsuffixed files
-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'
-N  do not search the standard directories for #include files
-n  emit code to check for dereferencing zero pointers
-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
-Name 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[pfblim]arg pass 'arg' to the preprocessor, compiler, assembler, bankpack, linker, ihxcheck, or makebin
```

## 12.2 sdcc settings

```
SDCC : mcs51/z80/z180/r2k/r2ka/r3ka/gbz80/tlcs90/ez80_z80/z80n/ds390/pic16/pic14/TININative/ds400/hc08/s08/stm8/pdk13/pdk14/pdk15/pdk16/pdk17/pdk18/pdk19/pdk20/pdk21/pdk22/pdk23/pdk24/pdk25/pdk26/pdk27/pdk28/pdk29/pdk30/pdk31/pdk32/pdk33/pdk34/pdk35/pdk36/pdk37/pdk38/pdk39/pdk40/pdk41/pdk42/pdk43/pdk44/pdk45/pdk46/pdk47/pdk48/pdk49/pdk50/pdk51/pdk52/pdk53/pdk54/pdk55/pdk56/pdk57/pdk58/pdk59/pdk60/pdk61/pdk62/pdk63/pdk64/pdk65/pdk66/pdk67/pdk68/pdk69/pdk70/pdk71/pdk72/pdk73/pdk74/pdk75/pdk76/pdk77/pdk78/pdk79/pdk80/pdk81/pdk82/pdk83/pdk84/pdk85/pdk86/pdk87/pdk88/pdk89/pdk90/pdk91/pdk92/pdk93/pdk94/pdk95/pdk96/pdk97/pdk98/pdk99/pdk100/pdk101/pdk102/pdk103/pdk104/pdk105/pdk106/pdk107/pdk108/pdk109/pdk110/pdk111/pdk112/pdk113/pdk114/pdk115/pdk116/pdk117/pdk118/pdk119/pdk120/pdk121/pdk122/pdk123/pdk124/pdk125/pdk126/pdk127/pdk128/pdk129/pdk130/pdk131/pdk132/pdk133/pdk134/pdk135/pdk136/pdk137/pdk138/pdk139/pdk140/pdk141/pdk142/pdk143/pdk144/pdk145/pdk146/pdk147/pdk148/pdk149/pdk150/pdk151/pdk152/pdk153/pdk154/pdk155/pdk156/pdk157/pdk158/pdk159/pdk160/pdk161/pdk162/pdk163/pdk164/pdk165/pdk166/pdk167/pdk168/pdk169/pdk170/pdk171/pdk172/pdk173/pdk174/pdk175/pdk176/pdk177/pdk178/pdk179/pdk180/pdk181/pdk182/pdk183/pdk184/pdk185/pdk186/pdk187/pdk188/pdk189/pdk190/pdk191/pdk192/pdk193/pdk194/pdk195/pdk196/pdk197/pdk198/pdk199/pdk200/pdk201/pdk202/pdk203/pdk204/pdk205/pdk206/pdk207/pdk208/pdk209/pdk210/pdk211/pdk212/pdk213/pdk214/pdk215/pdk216/pdk217/pdk218/pdk219/pdk220/pdk221/pdk222/pdk223/pdk224/pdk225/pdk226/pdk227/pdk228/pdk229/pdk230/pdk231/pdk232/pdk233/pdk234/pdk235/pdk236/pdk237/pdk238/pdk239/pdk240/pdk241/pdk242/pdk243/pdk244/pdk245/pdk246/pdk247/pdk248/pdk249/pdk250/pdk251/pdk252/pdk253/pdk254/pdk255/pdk256/pdk257/pdk258/pdk259/pdk260/pdk261/pdk262/pdk263/pdk264/pdk265/pdk266/pdk267/pdk268/pdk269/pdk270/pdk271/pdk272/pdk273/pdk274/pdk275/pdk276/pdk277/pdk278/pdk279/pdk280/pdk281/pdk282/pdk283/pdk284/pdk285/pdk286/pdk287/pdk288/pdk289/pdk290/pdk291/pdk292/pdk293/pdk294/pdk295/pdk296/pdk297/pdk298/pdk299/pdk300/pdk301/pdk302/pdk303/pdk304/pdk305/pdk306/pdk307/pdk308/pdk309/pdk310/pdk311/pdk312/pdk313/pdk314/pdk315/pdk316/pdk317/pdk318/pdk319/pdk320/pdk321/pdk322/pdk323/pdk324/pdk325/pdk326/pdk327/pdk328/pdk329/pdk330/pdk331/pdk332/pdk333/pdk334/pdk335/pdk336/pdk337/pdk338/pdk339/pdk340/pdk341/pdk342/pdk343/pdk344/pdk345/pdk346/pdk347/pdk348/pdk349/pdk350/pdk351/pdk352/pdk353/pdk354/pdk355/pdk356/pdk357/pdk358/pdk359/pdk360/pdk361/pdk362/pdk363/pdk364/pdk365/pdk366/pdk367/pdk368/pdk369/pdk370/pdk371/pdk372/pdk373/pdk374/pdk375/pdk376/pdk377/pdk378/pdk379/pdk380/pdk381/pdk382/pdk383/pdk384/pdk385/pdk386/pdk387/pdk388/pdk389/pdk390/pdk391/pdk392/pdk393/pdk394/pdk395/pdk396/pdk397/pdk398/pdk399/pdk400/pdk401/pdk402/pdk403/pdk404/pdk405/pdk406/pdk407/pdk408/pdk409/pdk410/pdk411/pdk412/pdk413/pdk414/pdk415/pdk416/pdk417/pdk418/pdk419/pdk420/pdk421/pdk422/pdk423/pdk424/pdk425/pdk426/pdk427/pdk428/pdk429/pdk430/pdk431/pdk432/pdk433/pdk434/pdk435/pdk436/pdk437/pdk438/pdk439/pdk440/pdk441/pdk442/pdk443/pdk444/pdk445/pdk446/pdk447/pdk448/pdk449/pdk450/pdk451/pdk452/pdk453/pdk454/pdk455/pdk456/pdk457/pdk458/pdk459/pdk460/pdk461/pdk462/pdk463/pdk464/pdk465/pdk466/pdk467/pdk468/pdk469/pdk470/pdk471/pdk472/pdk473/pdk474/pdk475/pdk476/pdk477/pdk478/pdk479/pdk480/pdk481/pdk482/pdk483/pdk484/pdk485/pdk486/pdk487/pdk488/pdk489/pdk490/pdk491/pdk492/pdk493/pdk494/pdk495/pdk496/pdk497/pdk498/pdk499/pdk500/pdk501/pdk502/pdk503/pdk504/pdk505/pdk506/pdk507/pdk508/pdk509/pdk510/pdk511/pdk512/pdk513/pdk514/pdk515/pdk516/pdk517/pdk518/pdk519/pdk520/pdk521/pdk522/pdk523/pdk524/pdk525/pdk526/pdk527/pdk528/pdk529/pdk530/pdk531/pdk532/pdk533/pdk534/pdk535/pdk536/pdk537/pdk538/pdk539/pdk540/pdk541/pdk542/pdk543/pdk544/pdk545/pdk546/pdk547/pdk548/pdk549/pdk550/pdk551/pdk552/pdk553/pdk554/pdk555/pdk556/pdk557/pdk558/pdk559/pdk560/pdk561/pdk562/pdk563/pdk564/pdk565/pdk566/pdk567/pdk568/pdk569/pdk570/pdk571/pdk572/pdk573/pdk574/pdk575/pdk576/pdk577/pdk578/pdk579/pdk580/pdk581/pdk582/pdk583/pdk584/pdk585/pdk586/pdk587/pdk588/pdk589/pdk590/pdk591/pdk592/pdk593/pdk594/pdk595/pdk596/pdk597/pdk598/pdk599/pdk600/pdk601/pdk602/pdk603/pdk604/pdk605/pdk606/pdk607/pdk608/pdk609/pdk610/pdk611/pdk612/pdk613/pdk614/pdk615/pdk616/pdk617/pdk618/pdk619/pdk620/pdk621/pdk622/pdk623/pdk624/pdk625/pdk626/pdk627/pdk628/pdk629/pdk630/pdk631/pdk632/pdk633/pdk634/pdk635/pdk636/pdk637/pdk638/pdk639/pdk640/pdk641/pdk642/pdk643/pdk644/pdk645/pdk646/pdk647/pdk648/pdk649/pdk650/pdk651/pdk652/pdk653/pdk654/pdk655/pdk656/pdk657/pdk658/pdk659/pdk660/pdk661/pdk662/pdk663/pdk664/pdk665/pdk666/pdk667/pdk668/pdk669/pdk670/pdk671/pdk672/pdk673/pdk674/pdk675/pdk676/pdk677/pdk678/pdk679/pdk680/pdk681/pdk682/pdk683/pdk684/pdk685/pdk686/pdk687/pdk688/pdk689/pdk690/pdk691/pdk692/pdk693/pdk694/pdk695/pdk696/pdk697/pdk698/pdk699/pdk700/pdk701/pdk702/pdk703/pdk704/pdk705/pdk706/pdk707/pdk708/pdk709/pdk710/pdk711/pdk712/pdk713/pdk714/pdk715/pdk716/pdk717/pdk718/pdk719/pdk720/pdk721/pdk722/pdk723/pdk724/pdk725/pdk726/pdk727/pdk728/pdk729/pdk730/pdk731/pdk732/pdk733/pdk734/pdk735/pdk736/pdk737/pdk738/pdk739/pdk740/pdk741/pdk742/pdk743/pdk744/pdk745/pdk746/pdk747/pdk748/pdk749/pdk750/pdk751/pdk752/pdk753/pdk754/pdk755/pdk756/pdk757/pdk758/pdk759/pdk760/pdk761/pdk762/pdk763/pdk764/pdk765/pdk766/pdk767/pdk768/pdk769/pdk770/pdk771/pdk772/pdk773/pdk774/pdk775/pdk776/pdk777/pdk778/pdk779/pdk780/pdk781/pdk782/pdk783/pdk784/pdk785/pdk786/pdk787/pdk788/pdk789/pdk790/pdk791/pdk792/pdk793/pdk794/pdk795/pdk796/pdk797/pdk798/pdk799/pdk800/pdk801/pdk802/pdk803/pdk804/pdk805/pdk806/pdk807/pdk808/pdk809/pdk810/pdk811/pdk812/pdk813/pdk814/pdk815/pdk816/pdk817/pdk818/pdk819/pdk820/pdk821/pdk822/pdk823/pdk824/pdk825/pdk826/pdk827/pdk828/pdk829/pdk830/pdk831/pdk832/pdk833/p
```

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                   Undefine macro as in -Umacro
-U                   Preprocessor option
-M                   Pass through options to the pre-processor (p), assembler (a) or linker (l)
-S                   Compile only; do not assemble or link
-c --compile-only     Compile and assemble, but do not link
-E --preprocessonly   Preprocess only, do not compile
--clmode             Act in cl 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$oft visual studio
--use-stdout         send errors to stdout instead of stderr
--nostdlib           Do not include the standard library directory in the search path
--nostdinc           Do not include the standard include directory in the search path
--less-pedantic      Disable some of the more pedantic warnings
--disable-warning    <nnnn> Disable specific warning
--Werror            Treat the warnings as errors
--debug             Enable debugging symbol output
--cyclomatic        Display complexity of compiled functions
--std-c89            Use ISO C90 (aka ANSI C89) standard (slightly incomplete)
--std-sdcc89        Use ISO C90 (aka ANSI C89) standard with SDCC extensions
--std-c95            Use ISO C95 (aka ISO C94) standard (slightly incomplete)
--std-c99            Use ISO C99 standard (incomplete)
--std-sdcc99        Use ISO C99 standard with SDCC extensions
--std-c11            Use ISO C11 standard (incomplete)
--std-sdcc11        Use ISO C11 standard with SDCC extensions (default)
--std-c2x            Use ISO C2X standard (incomplete)
--std-sdcc2x        Use ISO C2X standard with SDCC extensions
--fdollars-in-identifiers Permit '$' as an identifier character
--fsigned-char       Make "char" signed by default
--use-non-free       Search / include non-free licensed libraries and header files

```

```
Code generation options:
```

```
-m Set the port to use e.g. -mz80.
-p Select port specific processor e.g. -mpic14 -pl6f84
--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 caller <func[,func,...]> Cause the called function to save registers instead of the caller
--profile On supported ports, generate extra profiling information
--fomit-frame-pointer Leave out the frame pointer.
--all-callee-saves callee will always save registers used
--stack-probe insert call to function __stack_probe at each function prologue
--no-xinit-opt don't memcpy initialized xram from code
--no-c-code-in-asm don't include c-code as comments in the asm file
--no-peep-comments don't include peephole optimizer comments
--codeseg <name> use this name for the code segment
--constseg <name> use this name for the const segment
--dataseg <name> use this name for the data segment
```

Optimization options:

--nooverlay	Disable overlaying leaf function auto variables
--nogcse	Disable the GCSE optimisation
--nolabelopt	Disable label optimisation
--noinvariant	Disable optimisation of invariants
--noinduction	Disable loop variable induction
--noloopreverse	Disable the loop reverse optimisation
--no-peep	Disable the peephole assembly file optimisation
--no-reg-params	On some ports, disable passing some parameters in registers

```

--peep-asm           Enable peephole optimization on inline assembly
--peep-return        Enable peephole optimization for return instructions
--no-peep-return     Disable peephole optimization for return instructions
--peep-file          <file> use this extra peephole file
--opt-code-speed     Optimize for code speed rather than size
--opt-code-size      Optimize for code size rather than speed
--max-allocs-per-node Maximum number of register assignments considered at each node of the tree
                     decomposition
--nolospref          Disable lospref
--allow-unsafe-read  Allow optimizations to read any memory location anytime
--nostdlibcall       Disable optimization of calls to standard library

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

Linker options:
-l                  Include the given library in the link
-L                  Add the next field to the library search path
                     <path> use this path to search for libraries
--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 mcs51 port:
--model-small       internal data space is used (default)
--model-medium      external paged data space is used
--model-large       external data space is used
--model-huge        functions are banked, data in external space
--stack-size        Tells the linker to allocate this space for stack
--parms-in-bank1    use Bank1 for parameter passing
--acall-ajmp        Use acall/ajmp instead of lcall/ljmp
--no-ret-without-call Do not use ret independent of acall/lcall

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)
--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       For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy   Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc         Use old register allocator
--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

Special options for the z180 port:
--callee-saves-bc  Force a called function to always save BC
--portmode=         Determine PORT I/O mode (z80/z180)
--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       For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy   Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc         Use old register allocator
--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

Special options for the r2k port:
--callee-saves-bc  Force a called function to always save BC
--portmode=         Determine PORT I/O mode (z80/z180)
--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       For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy   Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc         Use old register allocator
--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

Special options for the r2ka port:
--callee-saves-bc  Force a called function to always save BC
--portmode=         Determine PORT I/O mode (z80/z180)

```

```

--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          For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy      Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc           Use old register allocator
--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

Special options for the r3ka port:
--callee-saves-bc     Force a called function to always save BC
--portmode=           Determine PORT I/O mode (z80/z180)
--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          For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy      Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc           Use old register allocator
--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

Special options for the gbz80 port:
--bo                  <num> use code bank <num>
--ba                  <num> use data bank <num>
--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          For the z80/gbz80 do not link default crt0.rel
--legacy-banking       Use legacy method to call banked functions

Special options for the tlcs90 port:
--callee-saves-bc     Force a called function to always save BC
--portmode=           Determine PORT I/O mode (z80/z180)
--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          For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy      Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc           Use old register allocator
--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

Special options for the ez80_z80 port:
--callee-saves-bc     Force a called function to always save BC
--portmode=           Determine PORT I/O mode (z80/z180)
--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          For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy      Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc           Use old register allocator
--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

Special options for the z80n port:
--callee-saves-bc     Force a called function to always save BC
--portmode=           Determine PORT I/O mode (z80/z180)
--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          For the z80/gbz80 do not link default crt0.rel
--reserve-regs-iy      Do not use IY (incompatible with --fomit-frame-pointer)
--oldralloc           Use old register allocator
--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

Special options for the ds390 port:
--model-flat24         use the flat24 model for the ds390 (default)
--stack-8bit           use the 8bit stack for the ds390 (not supported yet)
--stack-size           Tells the linker to allocate this space for stack
--stack-10bit          use the 10bit stack for ds390 (default)
--use-accelerator       generate code for ds390 arithmetic accelerator
--protect-sp-update    will disable interrupts during ESP:SP updates
--parms-in-bank1       use Bank1 for parameter passing

Special options for the pic16 port:
--pstack-model=        use stack model 'small' (default) or 'large'
-y --extended          enable Extended Instruction Set/Literal Offset Addressing mode

```

```

--pno-bankssel      do not generate BANKSEL assembler directives
--obankssel=        set bankssel optimization level (default=0 no)
--disable-peepts    explicit enable of peeptools
--no-optimize-goto   do NOT use (conditional) BRA instead of GOTO
--optimize-cmp       try to optimize some compares
--optimize-df        thoroughly analyze data flow (memory and time intensive!)
--asm=              Use alternative assembler
--mplab-comp         enable compatibility mode for MPLAB utilities (MPASM/MPLINK)
--link=              Use alternative linker
--preplace-udata-with= Place udata variables at another section: udata_acs, udata_ovr, udata_shr
--ivt-loc=           Set address of interrupt vector table.
--nodefaultlibs      do not link default libraries when linking
--use-crt=            use <crt-o> run-time initialization module
--no-crt             do not link any default run-time initialization module
--debug-xtra         show more debug info in assembly output
--debug-ralloc       dump register allocator debug file *.d
--pcode-verbose       dump pcode related info
--calltree           dump call tree in .calltree file
--gstack            trace stack pointer push/pop to overflow
--no-warn-non-free    suppress warning on absent --use-non-free option
Special options for the pic14 port:
--debug-xtra         show more debug info in assembly output
--no-pcode-opt       disable (slightly faulty) optimization on pCode
--stack-size         sets the size if the argument passing stack (default: 16, minimum: 4)
--no-extended-instructions forbid use of the extended instruction set (e.g., ADDFSR)
--no-warn-non-free    suppress warning on absent --use-non-free option
Special options for the TININative port:
--model-flat24       use the flat24 model for the ds390 (default)
--stack-8bit         use the 8bit stack for the ds390 (not supported yet)
--stack-size         Tells the linker to allocate this space for stack
--stack-10bit        use the 10bit stack for ds390 (default)
--use-accelerator     generate code for ds390 arithmetic accelerator
--protect-sp-update   will disable interrupts during ESP:SP updates
--parms-in-bank1     use Bank1 for parameter passing
--tini-libid         <nnnn> LibraryID used in -mTININative
Special options for the ds400 port:
--model-flat24       use the flat24 model for the ds400 (default)
--stack-8bit         use the 8bit stack for the ds400 (not supported yet)
--stack-size         Tells the linker to allocate this space for stack
--stack-10bit        use the 10bit stack for ds400 (default)
--use-accelerator     generate code for ds400 arithmetic accelerator
--protect-sp-update   will disable interrupts during ESP:SP updates
--parms-in-bank1     use Bank1 for parameter passing
Special options for the hc08 port:
--model-small        8-bit address space for data
--model-large        16-bit address space for data (default)
--out-fmt-elf        Output executable in ELF format
--oldralloc          Use old register allocator
Special options for the s08 port:
--model-small        8-bit address space for data
--model-large        16-bit address space for data (default)
--out-fmt-elf        Output executable in ELF format
--oldralloc          Use old register allocator
Special options for the stm8 port:
--model-medium       16-bit address space for both data and code (default)
--model-large        16-bit address space for data, 24-bit for code
--codeseg            <name> use this name for the code segment
--constseg           <name> use this name for the const segment
--out-fmt-elf        Output executable in ELF format

```

## 12.3 sdasgb settings

sdas Assembler V02.00 + NoICE + SDCC mods (GameBoy Z80-like CPU)

Copyright (C) 2012 Alan R. Baldwin

This program comes with ABSOLUTELY NO WARRANTY.

Usage: [-Options] file

Usage: [-Options] outfile file1 [file2 file3 ...]

```

-d    Decimal listing
-q    Octal listing
-x    Hex listing (default)
-g    Undefined symbols made global
-a    All user symbols made global
-b    Display .define substitutions in listing
-bb   and display without .define substitutions
-c    Disable instruction cycle count in listing
-j    Enable NoICE Debug Symbols
-y    Enable SDCC Debug Symbols
-l    Create list file/outfile[.lst]
-o    Create object file/outfile[.rel]
-s    Create symbol file/outfile[.sym]
-p    Disable automatic listing pagination
-u    Disable .list/.nlist processing
-w    Wide listing format for symbol table
-z    Disable case sensitivity for symbols
-f    Flag relocatable references by ' in listing file
-ff   Flag relocatable references by mode in listing file

```

```
-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.
removing
```

## 12.4 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
-yt<hexbyte> : Set MBC type per ROM byte 149 in Hex (see pandocs)
-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>"
-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>;
```

## 12.5 sdldgb settings

sdld Linker V03.00 + NoICE + 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
```

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
```

## 12.6 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"

## 12.7 makebin settings

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
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
  -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)
```

## 12.8 gbcompress settings

```
gbcompress [options] infile outfile
Use: Gbcompress a binary file and write it out.
Options
  -h : Show this help screen
  -d : Decompress (default is compress)
  -v : Verbose output
Example: "gbcompress binaryfile.bin compressed.bin"
Example: "gbcompress -d compressedfile.bin decompressed.bin"
```

## 12.9 png2mtspr settings

```
usage: png2mtspr <file>.png [options]
-c                ouput file (default: <png file>.c)
-sw <width>       metasprites width size (default: png width)
-sh <height>      metasprites height size (default: png height)
-px <x coord>     metasprites pivot x coordinate (default: metasprites width / 2)
-py <y coord>     metasprites pivot y coordinate (default: metasprites height / 2)
-spr8x8          use SPRITES_8x8 (default: SPRITES_8x16)
-spr8x16         use SPRITES_8x16 (default: SPRITES_8x16)
-b <bank>        bank (default 0)
```

# 13 Todo List

## Page [Coding Guidelines](#)

Update and verify this section for the modernized SDCC and toolchain

## File [far\\_ptr.h](#)

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

## Page [GBDK Toolchain](#)

Document png2mtspr

## File [malloc.h](#)

: This library may currently be broken.

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

Fill in this info for Banked Functions Banked functions (located in a switchable ROM bank)

- May call functions in any bank: ?
- May use data in any bank: **NO** (may only use data from currently active banks)

Const Data (Variables in ROM)

Variables in RAM

## Page [Using GBDK](#)

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

## 14 Module Index

### 14.1 C modules

Here is a list of all modules:

List of gbdk fonts	40
--------------------	----

## 15 Data Structure Index

### 15.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">__far_ptr</a>	41
<a href="#">_fixed</a>	42
<a href="#">atomic_flag</a>	42
<a href="#">joypads_t</a>	43
<a href="#">metasprite_t</a>	43
<a href="#">OAM_item_t</a>	44
<a href="#">sfont_handle</a>	45
<a href="#">smalloc_hunk</a>	45

## 16 File Index

### 16.1 File List

Here is a list of all files with brief descriptions:

<a href="#">assert.h</a>	51
<a href="#">bcd.h</a>	52
<a href="#">ctype.h</a>	54
<a href="#">gbdk-lib.h</a>	121
<a href="#">limits.h</a>	121
<a href="#">rand.h</a>	122
<a href="#">setjmp.h</a>	123
<a href="#">stdarg.h</a>	48
<a href="#">stdatomic.h</a>	125
<a href="#">stdbool.h</a>	125
<a href="#">stddef.h</a>	125
<a href="#">stdint.h</a>	126
<a href="#">stdio.h</a>	132



<a href="#">stdlib.h</a>	133
<a href="#">stdnoreturn.h</a>	137
<a href="#">string.h</a>	137
<a href="#">time.h</a>	141
<a href="#">typeof.h</a>	142
<a href="#">types.h</a>	51
<a href="#">asm/types.h</a>	50
<a href="#">asm/gbz80/provides.h</a>	47
<a href="#">asm/gbz80/stdarg.h</a>	48
<a href="#">asm/gbz80/types.h</a>	48
<a href="#">gb/bgb_emu.h</a>	55
<a href="#">gb/cgb.h</a>	57
<a href="#">gb/console.h</a>	62
<a href="#">gb/crash_handler.h</a>	63
<a href="#">gb/drawing.h</a>	63
<a href="#">gb/far_ptr.h</a>	68
<a href="#">gb/font.h</a>	72
<a href="#">gb/gb.h</a>	73
<a href="#">gb/gbdecompress.h</a>	106
<a href="#">gb/hardware.h</a>	108
<a href="#">gb/malloc.h</a>	112
<a href="#">gb/metasprites.h</a>	113
<a href="#">gb/sample.h</a>	117
<a href="#">gb/sgb.h</a>	118

## 17 Module Documentation

### 17.1 List of gbdk fonts

#### 17.1.1 Description

##### Variables

- [UINT8 font\\_spect \[\]](#)
- [UINT8 font\\_italic \[\]](#)
- [UINT8 font\\_ibm \[\]](#)
- [UINT8 font\\_min \[\]](#)
- [UINT8 font\\_ibm\\_fixed \[\]](#)

### 17.1.2 Variable Documentation

#### 17.1.2.1 font\_spect `UINT8 font_spect[]`

The default fonts

#### 17.1.2.2 font\_italic `UINT8 font_italic[]`

#### 17.1.2.3 font\_ibm `UINT8 font_ibm[]`

#### 17.1.2.4 font\_min `UINT8 font_min[]`

#### 17.1.2.5 font\_ibm\_fixed `UINT8 font_ibm_fixed[]`

Backwards compatible font

## 18 Data Structure Documentation

### 18.1 `__far_ptr` Union Reference

```
#include <far_ptr.h>
```

#### Data Fields

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

#### 18.1.1 Detailed Description

Union for working with members of a `FAR_PTR`

#### 18.1.2 Field Documentation

##### 18.1.2.1 `ptr` `FAR_PTR __far_ptr::ptr`

##### 18.1.2.2 `ofs` `void* __far_ptr::ofs`

##### 18.1.2.3 `seg` `unsigned int __far_ptr::seg`

**18.1.2.4 segofs** `struct { ... } __far_ptr::segofs`

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

**18.1.2.6 segfn** `struct { ... } __far_ptr::segfn`

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

- [gb/far\\_ptr.h](#)

## 18.2 \_fixed Union Reference

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

### Data Fields

- `struct {  
    UBYTE i  
    UBYTE h  
} b`
- `UWORD w`

### 18.2.1 Detailed Description

Useful definition for fixed point values

### 18.2.2 Field Documentation

**18.2.2.1 i** `UBYTE _fixed::i`

**18.2.2.2 h** `UBYTE _fixed::h`

**18.2.2.3 b** `struct { ... } _fixed::b`

**18.2.2.4 w** `UWORD _fixed::w`

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

- [asm/types.h](#)

## 18.3 atomic\_flag Struct Reference

```
#include <stdatomic.h>
```

### Data Fields

- unsigned char [flag](#)

### 18.3.1 Field Documentation

### 18.3.1.1 flag `unsigned char atomic_flag::flag`

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

- [stdatomic.h](#)

## 18.4 joypads\_t Struct Reference

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

### Data Fields

- [UINT8 npads](#)
- [union {  
    struct {  
        \[UINT8 joy0\]\(#\)  
        \[UINT8 joy1\]\(#\)  
        \[UINT8 joy2\]\(#\)  
        \[UINT8 joy3\]\(#\)  
    }  
    \[UINT8 joypads\]\(#\) \[4\]  
};](#)

### 18.4.1 Detailed Description

Multiplayer joypad structure.

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

### 18.4.2 Field Documentation

**18.4.2.1 npads** [UINT8](#) `joypads_t::npads`

**18.4.2.2 joy0** [UINT8](#) `joypads_t::joy0`

**18.4.2.3 joy1** [UINT8](#) `joypads_t::joy1`

**18.4.2.4 joy2** [UINT8](#) `joypads_t::joy2`

**18.4.2.5 joy3** [UINT8](#) `joypads_t::joy3`

**18.4.2.6 joypads** [UINT8](#) `joypads_t::joypads` [4]

**18.4.2.7 "@4 union { ... }**

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

- [gb/gb.h](#)

## 18.5 metasprite\_t Struct Reference

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

## Data Fields

- [INT8 dy](#)
- [INT8 dx](#)
- [UINT8 dtile](#)
- [UINT8 props](#)

### 18.5.1 Detailed Description

Metasprite sub-item structure

#### Parameters

<i>dy</i>	(INT8) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(INT8) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(UINT8) Start tile relative to the metasprites own set of tiles
<i>props</i>	(UINT8) 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).

### 18.5.2 Field Documentation

**18.5.2.1 dy** [INT8](#) `metasprite_t::dy`

**18.5.2.2 dx** [INT8](#) `metasprite_t::dx`

**18.5.2.3 dtile** [UINT8](#) `metasprite_t::dtile`

**18.5.2.4 props** [UINT8](#) `metasprite_t::props`

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

- [gb/metasprires.h](#)

## 18.6 OAM\_item\_t Struct Reference

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

## Data Fields

- [UINT8 y](#)
- [UINT8 x](#)
- [UINT8 tile](#)
- [UINT8 prop](#)

### 18.6.1 Detailed Description

Sprite Attributes structure

#### Parameters

<i>x</i>	X Coordinate of the sprite on screen
----------	--------------------------------------

## Parameters

<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> )

## 18.6.2 Field Documentation

**18.6.2.1** `y` `UINT8` `OAM_item_t::y`

**18.6.2.2** `x` `UINT8` `OAM_item_t::x`

**18.6.2.3** `tile` `UINT8` `OAM_item_t::tile`

**18.6.2.4** `prop` `UINT8` `OAM_item_t::prop`

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

- [gb/gb.h](#)

## 18.7 `sfont_handle` Struct Reference

```
#include <font.h>
```

## Data Fields

- `UINT8` `first_tile`
- `void *` `font`

### 18.7.1 Detailed Description

Font handle structure

### 18.7.2 Field Documentation

**18.7.2.1** `first_tile` `UINT8` `sfont_handle::first_tile`

First tile used for font

**18.7.2.2** `font` `void*` `sfont_handle::font`

Pointer to the base of the font

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

- [gb/font.h](#)

## 18.8 `smalloc_hunk` Struct Reference

```
#include <malloc.h>
```

## Data Fields

- [UBYTE](#) `magic`
- [pmmalloc\\_hunk](#) `next`
- [UWORD](#) `size`
- `int` [status](#)

### 18.8.1 Field Documentation

**18.8.1.1 magic** [UBYTE](#) `smalloc_hunk::magic`

**18.8.1.2 next** [pmmalloc\\_hunk](#) `smalloc_hunk::next`

**18.8.1.3 size** [UWORD](#) `smalloc_hunk::size`

**18.8.1.4 status** `int` `smalloc_hunk::status`

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

- `gb/malloc.h`

## 19 File Documentation

- 19.1 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/01\\_getting\\_started.md](#) File Reference
- 19.2 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/02\\_links\\_and\\_tools.md](#) File Reference
- 19.3 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/03\\_using\\_gbdk.md](#) File Reference
- 19.4 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/04\\_coding\\_↔guidelines.md](#) File Reference
- 19.5 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/05\\_banking\\_mbcx.md](#) File Reference
- 19.6 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/06\\_toolchain.md](#) File Reference
- 19.7 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/07\\_sample\\_↔programs.md](#) File Reference
- 19.8 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/08\\_faq.md](#) File Reference
- 19.9 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/09\\_migrating\\_new\\_↔versions.md](#) File Reference
- 19.10 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/10\\_release\\_notes.md](#) File Reference
- 19.11 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/20\\_toolchain\\_↔settings.md](#) File Reference
- 19.12 [/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/docs\\_index.md](#) File Reference

### 19.13 [asm/gbz80/provides.h](#) File Reference

#### Macros

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

#### 19.13.1 Macro Definition Documentation

19.13.1.1 **USE\_C\_MEMCPY** `#define USE_C_MEMCPY 0`

19.13.1.2 **USE\_C\_STRCPY** `#define USE_C_STRCPY 0`



**19.13.1.3 USE\_C\_STRCMP** `#define USE_C_STRCMP 0`

## 19.14 asm/gbz80/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`

### 19.14.1 Macro Definition Documentation

**19.14.1.1 va\_start** `#define va_start(  
list,  
last ) list = (unsigned char *)&last + sizeof(last)`

**19.14.1.2 va\_arg** `#define va_arg(  
list,  
type ) *((type *)((list += sizeof(type)) - sizeof(type)))`

**19.14.1.3 va\_end** `#define va_end(  
list )`

### 19.14.2 Typedef Documentation

**19.14.2.1 va\_list** `typedef unsigned char* va_list`

## 19.15 stdarg.h File Reference

`#include <asm/gbz80/stdarg.h>`

## 19.16 asm/gbz80/types.h File Reference

### Macros

- `#define NONBANKED __nonbanked`
- `#define BANKED __banked`
- `#define CRITICAL __critical`
- `#define INTERRUPT __interrupt`
- `#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 int [size\\_t](#)
- typedef [UINT16](#) [clock\\_t](#)

### 19.16.1 Detailed Description

Types definitions for the gb.

### 19.16.2 Macro Definition Documentation

**19.16.2.1 NONBANKED** `#define NONBANKED __nonbanked`

**19.16.2.2 BANKED** `#define BANKED __banked`

**19.16.2.3 CRITICAL** `#define CRITICAL __critical`

**19.16.2.4 INTERRUPT** `#define INTERRUPT __interrupt`

**19.16.2.5 \_\_SIZE\_T\_DEFINED** `#define __SIZE_T_DEFINED`

### 19.16.3 Typedef Documentation

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

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

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

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

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

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

**19.16.3.7 size\_t** typedef int [size\\_t](#)

**19.16.3.8 clock\_t** typedef [UINT16](#) [clock\\_t](#)  
Returned from clock

See also

[clock](#)

## 19.17 asm/types.h File Reference

```
#include <asm/gbz80/types.h>
```

### Data Structures

- union [\\_fixed](#)

### 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](#)

### 19.17.1 Detailed Description

Shared types definitions.

### 19.17.2 Typedef Documentation

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

**19.17.2.2 BYTE** typedef [INT8](#) [BYTE](#)  
Signed 8 bit.

**19.17.2.3 UBYTE** typedef [UINT8](#) [UBYTE](#)  
Unsigned 8 bit.

**19.17.2.4 WORD** typedef [INT16](#) [WORD](#)  
Signed 16 bit

**19.17.2.5 UWORD** typedef [UINT16](#) [UWORD](#)  
Unsigned 16 bit

**19.17.2.6 LWORD** typedef [INT32](#) [LWORD](#)  
Signed 32 bit

**19.17.2.7 ULWORD** typedef `UINT32 ULWORD`  
Unsigned 32 bit

**19.17.2.8 DWORD** typedef `INT32 DWORD`  
Signed 32 bit

**19.17.2.9 UDWORD** typedef `UINT32 UDWORD`  
Unsigned 32 bit

**19.17.2.10 fixed** typedef union `_fixed fixed`  
Useful definition for fixed point values

## 19.18 types.h File Reference

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

### Macros

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

### Typedefs

- typedef void \* `POINTER`

### 19.18.1 Detailed Description

Basic types.  
Directly include the port specific file.

### 19.18.2 Macro Definition Documentation

**19.18.2.1 NULL** #define `NULL` 0  
Good 'ol NULL.

**19.18.2.2 FALSE** #define `FALSE` 0  
A 'false' value.

**19.18.2.3 TRUE** #define `TRUE` 1  
A 'true' value.

### 19.18.3 Typedef Documentation

**19.18.3.1 POINTER** typedef void\* `POINTER`  
No longer used.

## 19.19 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)

### 19.19.1 Macro Definition Documentation

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

### 19.19.2 Function Documentation

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

## 19.20 bcd.h File Reference

`#include <asm/types.h>`

## Macros

- `#define BCD_HEX(v) ((BCD)(v))`
- `#define MAKE_BCD(v) BCD_HEX(0x ## v)`

## Typedefs

- `typedef unsigned long BCD`

## Functions

- void `uint2bcd` (unsigned int i, `BCD` \*value)
- void `bcd_add` (`BCD` \*sour, const `BCD` \*value)
- void `bcd_sub` (`BCD` \*sour, const `BCD` \*value)
- `UBYTE bcd2text` (const `BCD` \*bcd, `UBYTE` tile\_offset, unsigned char \*buffer)

### 19.20.1 Detailed Description

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

### 19.20.2 Macro Definition Documentation

**19.20.2.1 `BCD_HEX`** `#define BCD_HEX(`  
`v ) ((BCD)(v))`

**19.20.2.2 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

### 19.20.3 Typedef Documentation

**19.20.3.1 BCD** `typedef unsigned long BCD`

### 19.20.4 Function Documentation

**19.20.4.1 uint2bcd()** `void uint2bcd (  
    unsigned int 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

**19.20.4.2 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><i>sour</i></b>

**19.20.4.3 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><i>sour</i></b>

**19.20.4.4 bcd2text()** `UBYTE bcd2text (  
    const BCD * bcd,  
    UBYTE tile_offset,  
    unsigned char * 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](#).

## 19.21 ctype.h File Reference

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

**Functions**

- [BOOLEAN isalpha](#) (char *c*)
- [BOOLEAN isupper](#) (char *c*)
- [BOOLEAN islower](#) (char *c*)
- [BOOLEAN isdigit](#) (char *c*)
- [BOOLEAN isspace](#) (char *c*)
- char [toupper](#) (char *c*)
- char [tolower](#) (char *c*)

### 19.21.1 Detailed Description

Character type functions.

### 19.21.2 Function Documentation

**19.21.2.1 isalpha()** [BOOLEAN](#) isalpha (  
char *c* )

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

**Parameters**

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

**19.21.2.2 isupper()** [BOOLEAN](#) isupper (  
char *c* )

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

**Parameters**

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

**19.21.2.3 islower()** `BOOLEAN islower (char c)`

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

#### Parameters

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

**19.21.2.4 isdigit()** `BOOLEAN isdigit (char c)`

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

#### Parameters

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

**19.21.2.5 isspace()** `BOOLEAN isspace (char c)`

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

#### Parameters

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

**19.21.2.6 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

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

**19.21.2.7 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

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

## 19.22 gb/bgb\_emu.h File Reference

### Macros

- `#define BGB_MESSAGE(message_text) BGB_MESSAGE1(BGB_ADD_DOLLARD(__LINE__), message↵_text)`
- `#define BGB_MESSAGE_FMT(buf, ...) sprintf(buf, __VA_ARGS__);BGB_MESSAGE2(BGB_ADD_DOLL↵ARD(__LINE__), BGB_MAKE_LABEL(_##buf));`



- `#define BGB_PROFILE_BEGIN(MSG) BGB_MESSAGE(BGB_CONCAT(MSG,%ZEROCLKS%));`
- `#define BGB_PROFILE_END(MSG) BGB_MESSAGE(BGB_CONCAT(MSG,%-8+LASTCLKS%));`
- `#define BGB_TEXT(MSG) BGB_MESSAGE(BGB_STR(MSG))`

## Functions

- void `BGB_profiler_message()`

### 19.22.1 Detailed Description

Debug window logging and profiling support for the BGB emulator.

Also see the `bgb_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>

### 19.22.2 Macro Definition Documentation

**19.22.2.1 BGB\_MESSAGE** `#define BGB_MESSAGE( message_text ) BGB_MESSAGE1(BGB_ADD_DOLLARD(__LINE__), message_text)`

Macro to display a message in the BGB 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, LA←STCLKS, CLK2VBLANK

Example: print a message along with the currently active ROM bank.

```
BGB_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

[BGB\\_PROFILE\\_BEGIN\(\)](#), [BGB\\_PROFILE\\_END\(\)](#)

**19.22.2.2 BGB\_MESSAGE\_FMT** `#define BGB_MESSAGE_FMT( buf, ... ) sprintf(buf, __VA_ARGS__);BGB_MESSAGE2(BGB_ADD_DOLLARD(__LINE__), BGB_MA←KE_LABEL(_##buf));`

Macro to display a sprintf formatted message in the BGB emulator debug message window

#### Parameters

<i>buf</i>	Pointer to a globally defined char buffer
...	VA Args list of sprintf parameters

To avoid buffer overflows **buf** must be large enough to store the entire printed message.

Example:

```
char mybuf[100]; // should be globally defined
BGB_MESSAGE_FMT(mybuf, "An integer:%d, a string: %s", 12345, "hello bgb")
```

See also

[BGB\\_MESSAGE\(\)](#)

**19.22.2.3 BGB\_PROFILE\_BEGIN** `#define BGB_PROFILE_BEGIN(  
MSG ) BGB_MESSAGE (BGB_CONCAT (MSG, %ZEROCLKS%)) ;`

Macro to **Start** a profiling block for the BGB emulator

Parameters

<i>MSG</i>	Quoted text string to display in the debug message window
------------	---

To complete the profiling block and print the result call [BGB\\_PROFILE\\_END](#).

See also

[BGB\\_PROFILE\\_END\(\)](#), [BGB\\_MESSAGE\(\)](#)

**19.22.2.4 BGB\_PROFILE\_END** `#define BGB_PROFILE_END(  
MSG ) BGB_MESSAGE (BGB_CONCAT (MSG, %-8+LASTCLKS%)) ;`

Macro to **End** a profiling block and print the results in the BGB 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 [BGB\\_PROFILE\\_BEGIN\(\)](#)

The results are in BGB 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 ellapsed 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 ellapsed cycle count.

[BGB\\_MESSAGE](#) ("NOP TIME: %-4+LASTCLKS%");

See also

[BGB\\_PROFILE\\_BEGIN\(\)](#), [BGB\\_MESSAGE\(\)](#)

**19.22.2.5 BGB\_TEXT** `#define BGB_TEXT(  
MSG ) BGB_MESSAGE (BGB_STR (MSG))`

## 19.22.3 Function Documentation

**19.22.3.1 BGB\_profiler\_message()** `void BGB_profiler_message ( )`

Display preset debug information in the BGB debug messages window.

This function is equivalent to:

[BGB\\_MESSAGE](#) ("PROFILE, % (SP+\$0) %, % (SP+\$1) %, %A%, %TOTALCLKS%, %ROMBANK%, %WRAMBANK%");

## 19.23 gb/cgb.h File Reference

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

## Macros

- `#define RGB(r, g, b) (((UINT16)(b) & 0x1f) << 10) | (((UINT16)(g) & 0x1f) << 5) | (((UINT16)(r) & 0x1f) << 0))`
- `#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(11, 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)`

## Functions

- void `set_bkg_palette` (UINT8 first\_palette, UINT8 nb\_palettes, UINT16 \*rgb\_data) NONBANKED
- void `set_sprite_palette` (UINT8 first\_palette, UINT8 nb\_palettes, UINT16 \*rgb\_data) NONBANKED
- void `set_bkg_palette_entry` (UINT8 palette, UINT8 entry, UINT16 rgb\_data)
- void `set_sprite_palette_entry` (UINT8 palette, UINT8 entry, UINT16 rgb\_data)
- void `cpu_slow` (void)
- void `cpu_fast` (void)
- void `cgb_compatibility` (void)

### 19.23.1 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.

### 19.23.2 Macro Definition Documentation

**19.23.2.1 RGB** `#define RGB(  
    r,  
    g,  
    b ) (((UINT16)(b) & 0x1f) << 10) | (((UINT16)(g) & 0x1f) << 5) | (((UINT16)(r)  
& 0x1f) << 0))`

Macro to create a CGB palette color entry out of the color components.

## Parameters

<i>r</i>	Red Component, range 0 - 31 (31 brightest)
<i>g</i>	Green Component, range 0 - 31 (31 brightest)
<i>b</i>	Blue Component, range 0 - 31 (31 brightest)

The resulting format is BGR 15bpp.

See also

[set\\_bkg\\_palette\(\)](#), [set\\_sprite\\_palette\(\)](#)

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

Common colors based on the EGA default palette.

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

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

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

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

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

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

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

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

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

**19.23.2.12 RGB\_PINK** `#define RGB_PINK RGB(11, 0, 31)`

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

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

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

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

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

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

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

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

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

### 19.23.3 Function Documentation

**19.23.3.1 set\_bkg\_palette()** `void set_bkg_palette (`  
     `UINT8 first_palette,`  
     `UINT8 nb_palettes,`  
     `UINT16 * 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 15bpp 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\(\)](#)

**19.23.3.2 set\_sprite\_palette()** `void set_sprite_palette (`  
     `UINT8 first_palette,`  
     `UINT8 nb_palettes,`  
     `UINT16 * rgb_data )`

Set CGB sprite palette(s).

#### Parameters

<i>first_palette</i>	Index of the first palette to write (0-7)
<i>nb_palettes</i>	Number of palettes to write (1-8, max depends on first_palette)
<i>rgb_data</i>	Pointer to source palette data

Writes **nb\_palettes** to sprite palette data starting at **first\_palette**, Palette data is sourced from **rgb\_data**.

- Each Palette is 8 bytes in size: 4 colors x 2 bytes per palette color entry.
- Each color (4 per palette) is packed as BGR 15bpp 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\(\)](#)

**19.23.3.3 set\_bkg\_palette\_entry()** `void set_bkg_palette_entry (`  
`UINT8 palette,`  
`UINT8 entry,`  
`UINT16 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\(\)](#)

**19.23.3.4 set\_sprite\_palette\_entry()** `void set_sprite_palette_entry (`  
`UINT8 palette,`  
`UINT8 entry,`  
`UINT16 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\(\)](#)

**19.23.3.5 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\(\)](#)

**19.23.3.6 [cpu\\_fast\(\)](#)** `void cpu_fast (`  
`void )`

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](#)

**19.23.3.7 [cgb\\_compatibility\(\)](#)** `void cgb_compatibility (`  
`void )`

Set defaults compatible with the normal GameBoy models.

The default/first CGB palettes for sprites and backgrounds are set to a similar default appearance as on the DM↔G/Pocket/SGB models. (White, Light Gray, Dark Gray, Black)

- You can check to see if `_cpu == CGB_TYPE` before using this function.

## 19.24 [gb/console.h](#) File Reference

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

### Functions

- void [gotoxy](#) ([UINT8](#) x, [UINT8](#) y)
- [UINT8](#) [posx](#) (void)
- [UINT8](#) [posy](#) (void)
- void [setchar](#) (char c)
- void [cls](#) ()

#### 19.24.1 Detailed Description

Console functions that work like Turbo C's.

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

#### 19.24.2 Function Documentation

**19.24.2.1 [gotoxy\(\)](#)** `void gotoxy (`  
`UINT8 x,`  
`UINT8 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\(\)](#)

**19.24.2.2** `posx()` `UINT8 posx (`  
`void )`

Returns the current X position of the cursor.

See also

[gotoxy\(\)](#)

**19.24.2.3** `posy()` `UINT8 posy (`  
`void )`

Returns the current Y position of the cursor.

See also

[gotoxy\(\)](#)

**19.24.2.4** `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\(\)](#)

**19.24.2.5** `cls()` `void cls ( )`

Clears the screen

## 19.25 gb/crash\_handler.h File Reference

### Functions

- void [\\_\\_HandleCrash](#) ()

### 19.25.1 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.

### 19.25.2 Function Documentation

**19.25.2.1** `__HandleCrash()` `void __HandleCrash ( )`

Display the crash dump screen.

See the intro for this file for more details.

## 19.26 gb/drawing.h File Reference

`#include <types.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 number, INT8 radix, INT8 signed\_value)
- void `gprintn` (INT8 number, INT8 radix, INT8 signed\_value)
- INT8 `gprintf` (char \*fmt,...) **NONBANKED**
- void `plot` (UINT8 x, UINT8 y, UINT8 colour, UINT8 mode)
- void `plot_point` (UINT8 x, UINT8 y)
- void `switch_data` (UINT8 x, UINT8 y, unsigned char \*src, unsigned char \*dst) **NONBANKED**
- void `draw_image` (unsigned char \*data) **NONBANKED**
- void `line` (UINT8 x1, UINT8 y1, UINT8 x2, UINT8 y2)
- void `box` (UINT8 x1, UINT8 y1, UINT8 x2, UINT8 y2, UINT8 style)
- void `circle` (UINT8 x, UINT8 y, UINT8 radius, UINT8 style)
- UINT8 `getpix` (UINT8 x, UINT8 y)
- void `wrtchr` (char chr)
- void `gotogxy` (UINT8 x, UINT8 y)
- void `color` (UINT8 forecolor, UINT8 backcolor, UINT8 mode)

### 19.26.1 Detailed Description

All Points Addressable (APA) mode drawing library.

Drawing routines originally by Pascal Felber Legendary overhaul by Jon Fuge [jonny@q-continuum.demon.co.uk](mailto:jonny@q-continuum.demon.co.uk) 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.

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.

### 19.26.2 Macro Definition Documentation

#### 19.26.2.1 GRAPHICS\_WIDTH `#define GRAPHICS_WIDTH 160`

Size of the screen in pixels

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

**19.26.2.3 SOLID** `#define SOLID 0x00 /* Overwrites the existing pixels */`  
Possible drawing modes

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

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

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

**19.26.2.7 WHITE** `#define WHITE 0`  
Possible drawing colours

**19.26.2.8 LTGREY** `#define LTGREY 1`

**19.26.2.9 DKGREY** `#define DKGREY 2`

**19.26.2.10 BLACK** `#define BLACK 3`

**19.26.2.11 M\_NOFILL** `#define M_NOFILL 0`  
Possible fill styles for [box\(\)](#) and [circle\(\)](#)

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

**19.26.2.13 SIGNED** `#define SIGNED 1`  
Possible values for `signed_value` in [gprintln\(\)](#) and [gprintn\(\)](#)

**19.26.2.14 UNSIGNED** `#define UNSIGNED 0`

## 19.26.3 Function Documentation

**19.26.3.1 gprint()** `void gprint (`  
    `char * str )`

Print the string 'str' with no interpretation

See also

[gotogxy\(\)](#)

**19.26.3.2 gprintln()** `void gprintln (`  
    `INT16 number,`  
    `INT8 radix,`  
    `INT8 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 numer of characters printed.

See also

[gotogxy\(\)](#)

**19.26.3.3 gprintn()** `void gprintn (`  
    `INT8 number,`  
    `INT8 radix,`  
    `INT8 signed_value )`

Print 8 bit **number** in **radix** (base) in the default font at the current text position.

See also

[gprintln\(\)](#), [gotogxy\(\)](#)

**19.26.3.4 gprintf()** `INT8 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)
- %o (INT8 as octal)
- %x (INT8 as hex)
- %s (string)

**Returns**

Returns the number of items printed, or -1 if there was an error.

See also

[gotogxy\(\)](#)

**19.26.3.5 plot()** void plot (  
     UINT8 x,  
     UINT8 y,  
     UINT8 colour,  
     UINT8 mode )

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

**19.26.3.6 plot\_point()** void plot\_point (  
     UINT8 x,  
     UINT8 y )

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

**19.26.3.7 switch\_data()** void switch\_data (  
     UINT8 x,  
     UINT8 y,  
     unsigned char \* src,  
     unsigned char \* 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.

**19.26.3.8 draw\_image()** void draw\_image (  
     unsigned char \* data )

Draw a full screen image at **data**

**19.26.3.9 line()** void line (  
     UINT8 x1,  
     UINT8 y1,  
     UINT8 x2,  
     UINT8 y2 )

Draw a line in the current drawing mode and colour from **x1,y1** to **x2,y2**

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

Draw a box (rectangle) with corners **x1,y1** and **x2,y2** using fill mode **style** (one of NOFILL or FILL)

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

Draw a circle with centre at **x,y** and **radius** using fill mode **style** (one of NOFILL or FILL)

**19.26.3.12 getpix()** UINT8 getpix (  
     UINT8 x,  
     UINT8 y )

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

**19.26.3.13 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\(\)](#)

**19.26.3.14 gotogxy()** `void gotogxy (`  
    [UINT8](#) *x*,  
    [UINT8](#) *y* )

Sets the current text position to **x,y**.

Note: **x** and **y** have units of tiles (8 pixels per unit)

See also

[wrtchr\(\)](#)

**19.26.3.15 color()** `void color (`  
    [UINT8](#) *forecolor*,  
    [UINT8](#) *backcolor*,  
    [UINT8](#) *mode* )

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

## 19.27 gb/far\_ptr.h File Reference

### 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 unsigned long [FAR\\_PTR](#)

### Functions

- void [\\_\\_call\\_banked](#) ()
- long [to\\_far\\_ptr](#) (void \*ofs, int seg)

### Variables

- volatile [FAR\\_PTR \\_\\_call\\_banked\\_ptr](#)
- volatile void \* [\\_\\_call\\_banked\\_addr](#)
- volatile unsigned char [\\_\\_call\\_banked\\_bank](#)

#### 19.27.1 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)

## 19.27.2 Macro Definition Documentation

**19.27.2.1 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](#))

**19.27.2.2 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 <a href="#">FAR_PTR</a> )
------------	---

### Returns

Segment (Bank) of the far pointer (type unsigned int)

**19.27.2.3 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 <a href="#">FAR_PTR</a> )
------------	---

### Returns

Offset (address) of the far pointer (type void \*)

**19.27.2.4 FAR\_FUNC** `#define FAR_FUNC(  
    ptr,  
    typ ) ((typ) (((union __far_ptr *)&ptr)->segfn.fn))`

**19.27.2.5 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
int some_function(int param1, int param2) __banked { return 1; };
...
// Code elsewhere, such as unbanked main()
// This type declaration should match the above function
typedef int (*some_function_t)(int, int) __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)

## 19.27.3 Typedef Documentation

**19.27.3.1 FAR\_PTR** typedef unsigned long [FAR\\_PTR](#)  
Type for storing a FAR\_PTR

## 19.27.4 Function Documentation

**19.27.4.1 \_\_call\_banked()** void \_\_call\_banked ( )

**19.27.4.2 to\_far\_ptr()** long to\_far\_ptr (   
void \* ofs,   
int 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](#))

## 19.27.5 Variable Documentation

**19.27.5.1 \_\_call\_banked\_ptr** volatile [FAR\\_PTR](#) \_\_call\_banked\_ptr

**19.27.5.2 \_\_call\_banked\_addr** volatile void\* \_\_call\_banked\_addr

**19.27.5.3 \_\_call\_banked\_bank** volatile unsigned char \_\_call\_banked\_bank



## 19.28 gb/font.h File Reference

```
#include <gb/gb.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](#) [font\\_t](#)
- `typedef` struct [sfont\\_handle](#) [mfont\\_handle](#)
- `typedef` struct [sfont\\_handle](#) \* [pmfont\\_handle](#)

### Functions

- void [font\\_init](#) (void) [NONBANKED](#)
- [font\\_t](#) [font\\_load](#) (void \*font) [NONBANKED](#)
- [font\\_t](#) [font\\_set](#) ([font\\_t](#) font\_handle) [NONBANKED](#)

### Variables

- [UINT8](#) [font\\_spect](#) []
- [UINT8](#) [font\\_italic](#) []
- [UINT8](#) [font\\_ibm](#) []
- [UINT8](#) [font\\_min](#) []
- [UINT8](#) [font\\_ibm\\_fixed](#) []

#### 19.28.1 Detailed Description

Multiple font support for the GameBoy Michael Hope, 1999 [michaelh@earthling.net](mailto:michaelh@earthling.net)

#### 19.28.2 Macro Definition Documentation

**19.28.2.1 FONT\_256ENCODING** `#define FONT_256ENCODING 0`  
Various flags in the font header.

**19.28.2.2 FONT\_128ENCODING** `#define FONT_128ENCODING 1`

**19.28.2.3 FONT\_NOENCODING** `#define FONT_NOENCODING 2`

**19.28.2.4 FONT\_COMPRESSED** `#define FONT_COMPRESSED 4`

#### 19.28.3 Typedef Documentation

**19.28.3.1 font\_t** `typedef UINT16 font_t`

`font_t` is a handle to a font loaded by [font\\_load\(\)](#). It can be used with [font\\_set\(\)](#)

**19.28.3.2 mfont\_handle** `typedef struct sfont_handle mfont_handle`

Internal representation of a font. What a `font_t` really is

**19.28.3.3 pmfont\_handle** `typedef struct sfont_handle* pmfont_handle`**19.28.4 Function Documentation****19.28.4.1 font\_init()** `void font_init (`  
`void )`

Initializes the font system. Should be called before other font functions.

**19.28.4.2 font\_load()** `font_t font_load (`  
`void * font )`

Load a font and set it as the current font.

**Parameters**

<i>font</i>	Pointer to a font to load (usually a gbdk font)
-------------	---

**Returns**

Handle to the loaded font, which can be used with [font\\_set\(\)](#)

**See also**

[font\\_init\(\)](#), [font\\_set\(\)](#), [List of gbdk fonts](#)

**19.28.4.3 font\_set()** `font_t font_set (`  
`font_t font_handle )`

Set the current font.

**Parameters**

<i>font_handle</i>	handle of a font returned by <a href="#">font_load()</a>
--------------------	--

**Returns**

The previously used font handle.

**See also**

[font\\_init\(\)](#), [font\\_load\(\)](#)

**19.29 gb/gb.h File Reference**

```
#include <types.h>
#include <gb/hardware.h>
#include <gb/sgb.h>
#include <gb/cgb.h>
```

## Data Structures

- struct [joypads\\_t](#)
- struct [OAM\\_item\\_t](#)

## Macros

- #define [\\_\\_GBDK\\_VERSION](#) 402
- #define [J\\_START](#) 0x80U
- #define [J\\_SELECT](#) 0x40U
- #define [J\\_B](#) 0x20U
- #define [J\\_A](#) 0x10U
- #define [J\\_DOWN](#) 0x08U
- #define [J\\_UP](#) 0x04U
- #define [J\\_LEFT](#) 0x02U
- #define [J\\_RIGHT](#) 0x01U
- #define [M\\_DRAWING](#) 0x01U
- #define [M\\_TEXT\\_OUT](#) 0x02U
- #define [M\\_TEXT\\_INOUT](#) 0x03U
- #define [M\\_NO\\_SCROLL](#) 0x04U
- #define [M\\_NO\\_INTERP](#) 0x08U
- #define [S\\_PALETTE](#) 0x10U
- #define [S\\_FLIPX](#) 0x20U
- #define [S\\_FLIPY](#) 0x40U
- #define [S\\_PRIORITY](#) 0x80U
- #define [VBL\\_IFLAG](#) 0x01U
- #define [LCD\\_IFLAG](#) 0x02U
- #define [TIM\\_IFLAG](#) 0x04U
- #define [SIO\\_IFLAG](#) 0x08U
- #define [JOY\\_IFLAG](#) 0x10U
- #define [SCREENWIDTH](#) 0xA0U
- #define [SCREENHEIGHT](#) 0x90U
- #define [MINWNDPOSX](#) 0x07U
- #define [MINWNDPOSY](#) 0x00U
- #define [MAXWNDPOSX](#) 0xA6U
- #define [MAXWNDPOSY](#) 0x8FU
- #define [DMG\\_TYPE](#) 0x01
- #define [MGB\\_TYPE](#) 0xFF
- #define [CGB\\_TYPE](#) 0x11
- #define [IO\\_IDLE](#) 0x00U
- #define [IO\\_SENDING](#) 0x01U
- #define [IO\\_RECEIVING](#) 0x02U
- #define [IO\\_ERROR](#) 0x04U
- #define [SWITCH\\_ROM\\_MBC1](#)(b) [\\_current\\_bank](#) = (b), \*(unsigned char \*)0x2000 = (b)
- #define [SWITCH\\_RAM\\_MBC1](#)(b) \*(unsigned char \*)0x4000 = (b)
- #define [ENABLE\\_RAM\\_MBC1](#) \*(unsigned char \*)0x0000 = 0x0A
- #define [DISABLE\\_RAM\\_MBC1](#) \*(unsigned char \*)0x0000 = 0x00
- #define [SWITCH\\_16\\_8\\_MODE\\_MBC1](#) \*(unsigned char \*)0x6000 = 0x00
- #define [SWITCH\\_4\\_32\\_MODE\\_MBC1](#) \*(unsigned char \*)0x6000 = 0x01
- #define [SWITCH\\_ROM\\_MBC5](#)(b)
- #define [SWITCH\\_ROM\\_MBC5\\_8M](#)(b)
- #define [SWITCH\\_RAM\\_MBC5](#)(b) \*(unsigned char \*)0x4000 = (b)
- #define [ENABLE\\_RAM\\_MBC5](#) \*(unsigned char \*)0x0000 = 0x0A
- #define [DISABLE\\_RAM\\_MBC5](#) \*(unsigned char \*)0x0000 = 0x00
- #define [DISPLAY\\_ON](#) [LCDC\\_REG](#) |= 0x80U

- `#define DISPLAY_OFF display_off();`
- `#define SHOW_BKG LCDC_REG|=0x01U`
- `#define HIDE_BKG LCDC_REG&=0xFEU`
- `#define SHOW_WIN LCDC_REG|=0x20U`
- `#define HIDE_WIN LCDC_REG&=0xDFU`
- `#define SHOW_SPRITES LCDC_REG|=0x02U`
- `#define HIDE_SPRITES LCDC_REG&=0xFDU`
- `#define SPRITES_8x16 LCDC_REG|=0x04U`
- `#define SPRITES_8x8 LCDC_REG&=0xFBU`
- `#define DISABLE_OAM_DMA_shadow_OAM_base = 0`
- `#define ENABLE_OAM_DMA_shadow_OAM_base = (UBYTE)((UWORD)&shadow_OAM >> 8)`

## Typedefs

- `typedef void(* int_handler) (void) NONBANKED`
- `typedef struct OAM_item_t OAM_item_t`

## Functions

- `void remove_VBL (int_handler h) NONBANKED`
- `void remove_LCD (int_handler h) NONBANKED`
- `void remove_TIM (int_handler h) NONBANKED`
- `void remove_SIO (int_handler h) NONBANKED`
- `void remove_JOY (int_handler h) NONBANKED`
- `void add_VBL (int_handler h) NONBANKED`
- `void add_LCD (int_handler h) NONBANKED`
- `void add_TIM (int_handler h) NONBANKED`
- `void add_SIO (int_handler h) NONBANKED`
- `void add_JOY (int_handler h) NONBANKED`
- `void nowait_int_handler (void) NONBANKED`
- `void wait_int_handler (void) NONBANKED`
- `void mode (UINT8 m) NONBANKED`
- `UINT8 get_mode (void) NONBANKED __preserves_regs(b)`
- `void send_byte (void)`
- `void receive_byte (void)`
- `void delay (UINT16 d) NONBANKED`
- `UINT8 joypad (void) NONBANKED __preserves_regs(b)`
- `UINT8 waitpad (UINT8 mask) NONBANKED __preserves_regs(b)`
- `void waitpadup (void) NONBANKED __preserves_regs(a)`
- `UINT8 joypad_init (UINT8 npads, joypads_t *joypads)`
- `void joypad_ex (joypads_t *joypads)`
- `void enable_interrupts (void) NONBANKED __preserves_regs(a)`
- `void disable_interrupts (void) NONBANKED __preserves_regs(a)`
- `void set_interrupts (UINT8 flags) NONBANKED __preserves_regs(b)`
- `void reset (void) NONBANKED`
- `void wait_vbl_done (void) NONBANKED __preserves_regs(b)`
- `void display_off (void) NONBANKED __preserves_regs(b)`
- `void hiramcpy (UINT8 dst, const void *src, UINT8 n) NONBANKED __preserves_regs(b)`
- `void set_vram_byte (UBYTE *addr, UINT8 v) __preserves_regs(b)`
- `UINT8 * get_bkg_xy_addr (UINT8 x, UINT8 y) __preserves_regs(b)`
- `void set_bkg_data (UINT8 first_tile, UINT8 nb_tiles, const unsigned char *data) NONBANKED __preserves_regs(b)`
- `void set_bkg_1bit_data (UINT8 first_tile, UINT8 nb_tiles, const unsigned char *data, UINT8 color) NONBANKED __preserves_regs(b)`
- `void get_bkg_data (UINT8 first_tile, UINT8 nb_tiles, unsigned char *data) NONBANKED __preserves_regs(b)`

- void `set_bkg_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, const unsigned char \*tiles) NONBANKED \_\_preserves\_regs(b)↔
- void `set_bkg_submap` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, const unsigned char \*map, UINT8 map\_w)
- void `get_bkg_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char \*tiles) NONBANKED \_\_preserves\_regs(b)↔
- UINT8 \* `set_bkg_tile_xy` (UBYTE x, UBYTE y, UBYTE t) \_\_preserves\_regs(b)
- void `move_bkg` (UINT8 x, UINT8 y)
- void `scroll_bkg` (INT8 x, INT8 y)
- UINT8 \* `get_win_xy_addr` (UINT8 x, UINT8 y) \_\_preserves\_regs(b)
- void `set_win_data` (UINT8 first\_tile, UINT8 nb\_tiles, const unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `set_win_1bit_data` (UINT8 first\_tile, UINT8 nb\_tiles, const unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `get_win_data` (UINT8 first\_tile, UINT8 nb\_tiles, unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `set_win_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, const unsigned char \*tiles) NONBANKED \_\_preserves\_regs(b)↔
- void `set_win_submap` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, const unsigned char \*map, UINT8 map\_w)
- void `get_win_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char \*tiles) NONBANKED \_\_preserves\_regs(b)↔
- UINT8 \* `set_win_tile_xy` (UBYTE x, UBYTE y, UBYTE t) \_\_preserves\_regs(b)
- void `move_win` (UINT8 x, UINT8 y)
- void `scroll_win` (INT8 x, INT8 y)
- void `set_sprite_data` (UINT8 first\_tile, UINT8 nb\_tiles, const unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `set_sprite_1bit_data` (UINT8 first\_tile, UINT8 nb\_tiles, const unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `get_sprite_data` (UINT8 first\_tile, UINT8 nb\_tiles, unsigned char \*data) NONBANKED \_\_preserves\_regs(b)↔
- void `SET_SHADOW_OAM_ADDRESS` (void \*address)
- void `set_sprite_tile` (UINT8 nb, UINT8 tile)
- UINT8 `get_sprite_tile` (UINT8 nb)
- void `set_sprite_prop` (UINT8 nb, UINT8 prop)
- UINT8 `get_sprite_prop` (UINT8 nb)
- void `move_sprite` (UINT8 nb, UINT8 x, UINT8 y)
- void `scroll_sprite` (UINT8 nb, INT8 x, INT8 y)
- void `hide_sprite` (UINT8 nb)
- void `set_data` (unsigned char \*vram\_addr, const unsigned char \*data, UINT16 len) NONBANKED \_\_preserves\_regs(b)↔
- void `get_data` (unsigned char \*data, unsigned char \*vram\_addr, UINT16 len) NONBANKED \_\_preserves\_regs(b)↔
- void `set_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char \*vram\_addr, const unsigned char \*tiles) NONBANKED \_\_preserves\_regs(b)↔
- void `get_tiles` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char \*tiles, unsigned char \*vram\_addr) NONBANKED \_\_preserves\_regs(b)↔
- void `init_win` (UINT8 c) NONBANKED \_\_preserves\_regs(b)
- void `init_bkg` (UINT8 c) NONBANKED \_\_preserves\_regs(b)
- void `vmemset` (void \*s, UINT8 c, size\_t n) NONBANKED \_\_preserves\_regs(b)
- void `fill_bkg_rect` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, UINT8 tile) NONBANKED \_\_preserves\_regs(b)↔
- void `fill_win_rect` (UINT8 x, UINT8 y, UINT8 w, UINT8 h, UINT8 tile) NONBANKED \_\_preserves\_regs(b)↔

## Variables

- [UINT8 c](#)
- [UINT8 \\_cpu](#)
- volatile [UINT16 sys\\_time](#)
- volatile [UINT8 \\_io\\_status](#)
- volatile [UINT8 \\_io\\_in](#)
- volatile [UINT8 \\_io\\_out](#)
- [\\_\\_REG \\_current\\_bank](#)
- [UINT8 h](#)
- [UINT8 l](#)
- void [b](#)
- void [d](#)
- void [e](#)
- volatile struct [OAM\\_item\\_t shadow\\_OAM](#) []
- [\\_\\_REG \\_shadow\\_OAM\\_base](#)

### 19.29.1 Detailed Description

Gameboy specific functions.

### 19.29.2 Macro Definition Documentation

**19.29.2.1** [\\_\\_GBDK\\_VERSION](#) `#define __GBDK_VERSION 402`

**19.29.2.2** [J\\_START](#) `#define J_START 0x80U`

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

UINT8 keys; keys = [joypad\(\)](#); if (keys & J\_B) { ... }

See also

[joypad](#)

**19.29.2.3** [J\\_SELECT](#) `#define J_SELECT 0x40U`

**19.29.2.4** [J\\_B](#) `#define J_B 0x20U`

**19.29.2.5** [J\\_A](#) `#define J_A 0x10U`

**19.29.2.6** [J\\_DOWN](#) `#define J_DOWN 0x08U`

**19.29.2.7** [J\\_UP](#) `#define J_UP 0x04U`

**19.29.2.8** [J\\_LEFT](#) `#define J_LEFT 0x02U`

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

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

Screen modes. Normally used by internal functions only.

See also

[mode\(\)](#)

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

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

**19.29.2.13 M\_NO\_SCROLL** `#define M_NO_SCROLL 0x04U`

Set this in addition to the others to disable scrolling

If scrolling is disabled, the cursor returns to (0,0)

See also

[mode\(\)](#)

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

Set this to disable interpretation

See also

[mode\(\)](#)

**19.29.2.15 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\(\)](#).

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

If set the sprite will be flipped horizontally.

See also

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

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

If set the sprite will be flipped vertically.

See also

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

**19.29.2.18 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\(\)](#)

**19.29.2.19 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](#)

**19.29.2.20 LCD\_IFLAG** `#define LCD_IFLAG 0x02U`

LCD Interrupt when triggered by the STAT register.

See also

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

**19.29.2.21 TIM\_IFLAG** `#define TIM_IFLAG 0x04U`

Timer Interrupt when the timer [TIMA\\_REG](#) overflows.

See also

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

**19.29.2.22 SIO\_IFLAG** `#define SIO_IFLAG 0x08U`

Serial Link Interrupt occurs when the serial transfer has completed.

See also

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

**19.29.2.23 JOY\_IFLAG** `#define JOY_IFLAG 0x10U`

Joypad Interrupt occurs on a transition of the keypad.

See also

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

**19.29.2.24 SCREENWIDTH** `#define SCREENWIDTH 0xA0U`

Width of the visible screen in pixels.



**19.29.2.25 SCREENHEIGHT** `#define SCREENHEIGHT 0x90U`  
Height of the visible screen in pixels.

**19.29.2.26 MINWNDPOSX** `#define MINWNDPOSX 0x07U`  
The Minimum X position of the Window Layer (Left edge of screen)

See also

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

**19.29.2.27 MINWNDPOSY** `#define MINWNDPOSY 0x00U`  
The Minimum Y position of the Window Layer (Top edge of screen)

See also

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

**19.29.2.28 MAXWNDPOSX** `#define MAXWNDPOSX 0xA6U`  
The Maximum X position of the Window Layer (Right edge of screen)

See also

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

**19.29.2.29 MAXWNDPOSY** `#define MAXWNDPOSY 0x8FU`  
The Maximum Y position of the Window Layer (Bottom edge of screen)

See also

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

**19.29.2.30 DMG\_TYPE** `#define DMG_TYPE 0x01`  
Hardware Model: Original GB or Super GB.

See also

[\\_cpu](#)

**19.29.2.31 MGB\_TYPE** `#define MGB_TYPE 0xFF`  
Hardware Model: Pocket GB or Super GB 2.

See also

[\\_cpu](#)

**19.29.2.32 CGB\_TYPE** `#define CGB_TYPE 0x11`  
Hardware Model: Color GB.

See also

[\\_cpu](#)

**19.29.2.33 IO\_IDLE** `#define IO_IDLE 0x00U`  
Serial Link IO is completed

**19.29.2.34 IO\_SENDING** `#define IO_SENDING 0x01U`  
Serial Link Sending data

**19.29.2.35 IO\_RECEIVING** `#define IO_RECEIVING 0x02U`  
Serial Link Receiving data

**19.29.2.36 IO\_ERROR** `#define IO_ERROR 0x04U`  
Serial Link Error

**19.29.2.37 SWITCH\_ROM\_MBC1** `#define SWITCH_ROM_MBC1(  
    b ) _current_bank = (b), *(unsigned char *)0x2000 = (b)`  
Makes MBC1 and other compatible MBCs switch the active ROM bank

#### Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

**19.29.2.38 SWITCH\_RAM\_MBC1** `#define SWITCH_RAM_MBC1(  
    b ) *(unsigned char *)0x4000 = (b)`  
Switches SRAM bank on MBC1 and other compatible MBCs

#### Parameters

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

**19.29.2.39 ENABLE\_RAM\_MBC1** `#define ENABLE_RAM_MBC1 *(unsigned char *)0x0000 = 0x0A`  
Enables SRAM on MBC1

**19.29.2.40 DISABLE\_RAM\_MBC1** `#define DISABLE_RAM_MBC1 *(unsigned char *)0x0000 = 0x00`  
Disables SRAM on MBC1

**19.29.2.41 SWITCH\_16\_8\_MODE\_MBC1** `#define SWITCH_16_8_MODE_MBC1 *(unsigned char *)0x6000 = 0x00`

**19.29.2.42 SWITCH\_4\_32\_MODE\_MBC1** `#define SWITCH_4_32_MODE_MBC1 *(unsigned char *)0x6000 = 0x01`

**19.29.2.43 SWITCH\_ROM\_MBC5** `#define SWITCH_ROM_MBC5(  
    b )`

#### Value:

```
_current_bank = (b), \  
*(unsigned char *)0x3000 = 0, \  
*(unsigned char *)0x2000 = (b)
```

Makes MBC5 switch to the active ROM bank; only 4M roms are supported,

See also

[SWITCH\\_ROM\\_MBC5\\_8M\(\)](#)

## Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

Note the order used here. Writing the other way around on a MBC1 always selects bank 1

**19.29.2.44 SWITCH\_ROM\_MBC5\_8M** `#define SWITCH_ROM_MBC5_8M(  
    b )`

## Value:

```
*(unsigned char *)0x3000 = ((UINT16)(b) >> 8), \
*(unsigned char *)0x2000 = (b)
```

Makes MBC5 to switch the active ROM bank; active bank number is not tracked by `_current_bank` if you use this macro

## See also

[\\_current\\_bank](#)

## Parameters

<i>b</i>	ROM bank to switch to
----------	-----------------------

Note the order used here. Writing the other way around on a MBC1 always selects bank 1

**19.29.2.45 SWITCH\_RAM\_MBC5** `#define SWITCH_RAM_MBC5(  
    b )` `*(unsigned char *)0x4000 = (b)`

Switches SRAM bank on MBC5

## Parameters

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

**19.29.2.46 ENABLE\_RAM\_MBC5** `#define ENABLE_RAM_MBC5` `*(unsigned char *)0x0000 = 0x0A`  
Enables SRAM on MBC5

**19.29.2.47 DISABLE\_RAM\_MBC5** `#define DISABLE_RAM_MBC5` `*(unsigned char *)0x0000 = 0x00`  
Disables SRAM on MBC5

**19.29.2.48 DISPLAY\_ON** `#define DISPLAY_ON` `LCDC_REG|=0x80U`  
Turns the display back on.

## See also

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

**19.29.2.49 DISPLAY\_OFF** `#define DISPLAY_OFF` `display_off();`  
Turns the display off immediately.

## See also

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

**19.29.2.50 SHOW\_BKG** `#define SHOW_BKG` `LCDC_REG|=0x01U`  
Turns on the background layer. Sets bit 0 of the LCDC register to 1.

**19.29.2.51 HIDE\_BKG** `#define HIDE_BKG LCDC_REG&=0xFEU`  
Turns off the background layer. Sets bit 0 of the LCDC register to 0.

**19.29.2.52 SHOW\_WIN** `#define SHOW_WIN LCDC_REG|=0x20U`  
Turns on the window layer Sets bit 5 of the LCDC register to 1.

**19.29.2.53 HIDE\_WIN** `#define HIDE_WIN LCDC_REG&=0xDFU`  
Turns off the window layer. Clears bit 5 of the LCDC register to 0.

**19.29.2.54 SHOW\_SPRITES** `#define SHOW_SPRITES LCDC_REG|=0x02U`  
Turns on the sprites layer. Sets bit 1 of the LCDC register to 1.

**19.29.2.55 HIDE\_SPRITES** `#define HIDE_SPRITES LCDC_REG&=0xFDU`  
Turns off the sprites layer. Clears bit 1 of the LCDC register to 0.

**19.29.2.56 SPRITES\_8x16** `#define SPRITES_8x16 LCDC_REG|=0x04U`  
Sets sprite size to 8x16 pixels, two tiles one above the other. Sets bit 2 of the LCDC register to 1.

**19.29.2.57 SPRITES\_8x8** `#define SPRITES_8x8 LCDC_REG&=0xFBU`  
Sets sprite size to 8x8 pixels, one tile. Clears bit 2 of the LCDC register to 0.

**19.29.2.58 DISABLE\_OAM\_DMA** `#define DISABLE_OAM_DMA _shadow_OAM_base = 0`  
Disable OAM DMA copy each VBlank

**19.29.2.59 ENABLE\_OAM\_DMA** `#define ENABLE_OAM_DMA _shadow_OAM_base = (UBYTE)((UWORD)&shadow_OAM >> 8)`  
Enable OAM DMA copy each VBlank and set it to transfer default shadow\_OAM array

### 19.29.3 Typedef Documentation

**19.29.3.1 int\_handler** `typedef void(* int_handler) (void) NONBANKED`  
Interrupt handlers

**19.29.3.2 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> )

### 19.29.4 Function Documentation

**19.29.4.1 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\(\)](#)

**19.29.4.2 remove\_LCD()** `void remove_LCD (`  
`int_handler h )`

Removes the LCD interrupt handler.

See also

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

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

Removes the TIM interrupt handler.

See also

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

**19.29.4.4 remove\_SIO()** `void remove_SIO (`  
`int_handler h )`

Removes the Serial Link / SIO interrupt handler.

See also

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

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

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

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

Removes the JOY interrupt handler.

See also

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

**19.29.4.6 add\_VBL()** `void add_VBL (`  
`int_handler h )`

Adds a V-blank 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. If the [remove\\_VBL](#) function is to be called, only three may be added.

Note: The default VBL is installed automatically.

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

Adds a LCD interrupt handler.

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

From pan/k0Pa: There are various reasons for this interrupt to occur as described by the `STAT_REG` register (\$FF41). One very popular reason is to indicate to the user when the video hardware is about to redraw a given LCD line. This can be useful for dynamically controlling the `SCX_REG` / `SCY_REG` registers (\$FF43/\$FF42) to perform special video effects.

See also

[add\\_VBL](#)

**19.29.4.8 add\_TIM()** `void add_TIM (`  
     `int_handler h )`

Adds a timer interrupt handler.

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

See also

[add\\_VBL](#)

[set\\_interrupts\(\)](#) with `TIM_IFLAG`

**19.29.4.9 add\_SIO()** `void add_SIO (`  
     `int_handler h )`

Adds a Serial Link transmit complete interrupt handler.

From pan/k0Pa: This interrupt occurs when a serial transfer has completed on the game link port.

See also

[send\\_byte](#), [receive\\_byte\(\)](#), [add\\_VBL\(\)](#)

[set\\_interrupts\(\)](#) with `SIO_IFLAG`

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

Adds a joystick button change interrupt handler.

From pan/k0Pa: This interrupt occurs on a transition of any of the keypad input lines from high to low. Due to the fact that keypad "bounce" is virtually always present, software should expect this interrupt to occur one or more times for every button press and one or more times for every button release.

See also

[joypad\(\)](#)

**19.29.4.11 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\(\)](#)

**19.29.4.12 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\(\)](#)

**19.29.4.13 mode()** `void mode (`  
    `UINT8 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](#)

**19.29.4.14 get\_mode()** `UINT8 get_mode (`  
    `void )`

Returns the current mode

See also

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

**19.29.4.15 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](#)

**19.29.4.16 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](#)

**19.29.4.17 delay()** `void delay (`  
     `UINT16 d )`

Delays the given number of milliseconds. Uses no timers or interrupts, and can be called with interrupts disabled (why nobody knows :)

**19.29.4.18 joypad()** `UINT8 joypad (`  
     `void )`

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

See also

[J\\_START](#), [J\\_SELECT](#), [J\\_A](#), [J\\_B](#), [J\\_UP](#), [J\\_DOWN](#), [J\\_LEFT](#), [J\\_RIGHT](#)

**19.29.4.19 waitpad()** `UINT8 waitpad (`  
     `UINT8 mask )`

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

Parameters

<i>mask</i>	Bitmask indicating which buttons to wait for
-------------	--

Normally only used for checking one key, but it will support many, even J\_LEFT at the same time as J\_RIGHT. :)

Note: Checks in a loop that doesn't HALT at all, so the CPU will be maxed out until this call returns.

See also

[joypad](#)

[J\\_START](#), [J\\_SELECT](#), [J\\_A](#), [J\\_B](#), [J\\_UP](#), [J\\_DOWN](#), [J\\_LEFT](#), [J\\_RIGHT](#)

**19.29.4.20 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.

**19.29.4.21 joypad\_init()** `UINT8 joypad_init (`  
     `UINT8 npads,`  
     `joypads_t * joypads )`

Initializes [joypads\\_t](#) structure for polling multiple joypads (for the GB and ones connected via SGB)

Parameters

<i>npads</i>	number of joypads requested (1, 2 or 4)
<i>joypads</i>	pointer to <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](#)



**19.29.4.22 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](#)

**19.29.4.23 enable\_interrupts()** `void enable_interrupts (`  
     `void )`

Enables unmasked interrupts

#### See also

[disable\\_interrupts](#), [set\\_interrupts](#)

**19.29.4.24 disable\_interrupts()** `void disable_interrupts (`  
     `void )`

Disables interrupts.

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

#### See also

[enable\\_interrupts](#), [set\\_interrupts](#)

**19.29.4.25 set\_interrupts()** `void set_interrupts (`  
     `UINT8 flags )`

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

#### Parameters

<i>flags</i>	A logical OR of *_IFLAGS
--------------	--------------------------

#### See also

[enable\\_interrupts\(\)](#), [disable\\_interrupts\(\)](#)  
[VBL\\_IFLAG](#), [LCD\\_IFLAG](#), [TIM\\_IFLAG](#), [SIO\\_IFLAG](#), [JOY\\_IFLAG](#)

**19.29.4.26 reset()** `void reset (`  
     `void )`

Performs a warm reset by reloading the CPU value then jumping to the start of crt0 (0x0150)

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

HALTs the CPU and waits for the vertical blank interrupt (VBL) to finish.

This is often used in main loops to idle the CPU at low power until it's time to start the next frame. It's also useful for syncing animation with the screen re-draw.

Warning: If the VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediately.

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

Turns the display off.

Waits until the VBL interrupt before turning the display off.

See also

[DISPLAY\\_ON](#)

**19.29.4.29 hiramcpy()** `void hiramcpy (`  
    `UINT8 dst,`  
    `const void * src,`  
    `UINT8 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.

**19.29.4.30 set\_vram\_byte()** `void set_vram_byte (`  
    `UBYTE * addr,`  
    `UINT8 v )`

Set byte in vram at given memory location

Parameters

<i>addr</i>	address to write to
<i>v</i>	value

**19.29.4.31 get\_bkg\_xy\_addr()** `UINT8* get_bkg_xy_addr (`  
    `UINT8 x,`  
    `UINT8 y )`

Get address of X,Y tile of background map

**19.29.4.32 set\_bkg\_data()** `void set_bkg_data (`  
    `UINT8 first_tile,`  
    `UINT8 nb_tiles,`  
    `const unsigned char * 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

## Parameters

<i>data</i>	Pointer to (2 bpp) source tile data
-------------	-------------------------------------

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK\\_REG](#) determines which bank of Background tile patterns are written to.

- VBK\_REG=0 indicates the first bank
- VBK\_REG=1 indicates the second

**19.29.4.33 set\_bkg\_1bit\_data()** `void set_bkg_1bit_data (`  
     `UINT8 first_tile,`  
     `UINT8 nb_tiles,`  
     `const unsigned char * data,`  
     `UINT8 color )`

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
<i>color</i>	Color

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](#)

**19.29.4.34 get\_bkg\_data()** `void get_bkg_data (`  
     `UINT8 first_tile,`  
     `UINT8 nb_tiles,`  
     `unsigned char * data )`

Copies from Background / Window VRAM Tile Pattern data into a buffer

## Parameters

<i>first_tile</i>	Index of the first Tile to read from
<i>nb_tiles</i>	Number of Tiles to read
<i>data</i>	Pointer to destination buffer for Tile Pattern data

Copies **nb\_tiles** tiles from VRAM starting at **first\_tile**, Tile data is copied into **data**.

Each Tile is 16 bytes, so the buffer pointed to by **data** should be at least **nb\_tiles** x 16 bytes in size.

See also

[get\\_win\\_data](#)

**19.29.4.35 set\_bkg\_tiles()** `void set_bkg_tiles (`  
     `UINT8 x,`  
     `UINT8 y,`  
     `UINT8 w,`  
     `UINT8 h,`  
     `const unsigned char * 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=0 Tile Numbers are written
- VBK\_REG=1 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](#)

**19.29.4.36 set\_bkg\_submap()** `void set_bkg_submap (`  
     `UINT8 x,`  
     `UINT8 y,`  
     `UINT8 w,`  
     `UINT8 h,`  
     `const unsigned char * map,`  
     `UINT8 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 32 tiles.

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</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Background Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map\_w** as the rowstride for the source tile map.

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](#)

**19.29.4.37 get\_bkg\_tiles()** `void get_bkg_tiles (`  
     `UINT8 x,`  
     `UINT8 y,`  
     `UINT8 w,`  
     `UINT8 h,`  
     `unsigned char * tiles )`

Copies a rectangular region of Background Tile Map entries into a buffer.

Parameters

<i>x</i>	X Start position in Background Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Background Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Background Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x** x **y** bytes in size.

**19.29.4.38 set\_bkg\_tile\_xy()** `UINT8* set_bkg_tile_xy (`  
     `UBYTE x,`  
     `UBYTE y,`  
     `UBYTE 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

**19.29.4.39 move\_bkg()** `void move_bkg (`  
     `UINT8 x,`  
     `UINT8 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](#)

**19.29.4.40 scroll\_bkg()** `void scroll_bkg (`  
     `INT8 x,`  
     `INT8 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> Range: -128 - 127
<i>y</i>	Number of pixels to move the Background on the <b>Y axis</b> Range: -128 - 127

See also

[move\\_bkg](#)

**19.29.4.41** `get_win_xy_addr()` `UINT8* get_win_xy_addr (`  
    `UINT8 x,`  
    `UINT8 y )`

Get address of X,Y tile of window map

**19.29.4.42** `set_win_data()` `void set_win_data (`  
    `UINT8 first_tile,`  
    `UINT8 nb_tiles,`  
    `const unsigned char * 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](#)

[SHOW\\_WIN](#), [HIDE\\_WIN](#)

**19.29.4.43** `set_win_1bit_data()` `void set_win_1bit_data (`  
    `UINT8 first_tile,`  
    `UINT8 nb_tiles,`  
    `const unsigned char * 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\\_1bit\\_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

See also

[set\\_bkg\\_data](#), [set\\_bkg\\_1bit\\_data](#), [set\\_win\\_data](#)

**19.29.4.44** `get_win_data()` `void get_win_data (`  
    `UINT8 first_tile,`  
    `UINT8 nb_tiles,`  
    `unsigned char * data )`

Copies from Window / Background VRAM Tile Pattern data into a buffer

## Parameters

<i>first_tile</i>	Index of the first Tile to read from
<i>nb_tiles</i>	Number of Tiles to read
<i>data</i>	Pointer to destination buffer for Tile Pattern Data

This is the same as [get\\_bkg\\_data](#), since the Window Layer and Background Layer share the same Tile pattern data.

See also

[get\\_bkg\\_data](#)

**19.29.4.45 set\_win\_tiles()** `void set_win_tiles (`  
     **UINT8** *x*,  
     **UINT8** *y*,  
     **UINT8** *w*,  
     **UINT8** *h*,  
     const unsigned char \* *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=0 Tile Numbers are written
- VBK\_REG=1 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](#)

**19.29.4.46 set\_win\_submap()** `void set_win_submap (`  
     **UINT8** *x*,  
     **UINT8** *y*,  
     **UINT8** *w*,  
     **UINT8** *h*,



```
const unsigned char * map,  
UINT8 map_w )
```

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 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 - 255
<i>h</i>	Height of area to set in tiles. Range 1 - 255
<i>map</i>	Pointer to source tile map data
<i>map</i> ↔ <i>_w</i>	Width of source tile map in tiles. Range 1 - 255

Entries are copied from **map** to the Window Tile Map starting at **x**, **y** writing across for **w** tiles and down for **h** tiles, using **map\_w** as the rowstride for the source tile map.

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](#)=0 Tile Numbers are written
- [VBK\\_REG](#)=1 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](#)

**19.29.4.47** [get\\_win\\_tiles\(\)](#) `void get_win_tiles (`  
     [UINT8](#) *x*,  
     [UINT8](#) *y*,  
     [UINT8](#) *w*,  
     [UINT8](#) *h*,  
     unsigned char \* *tiles* )

Copies a rectangular region of Window Tile Map entries into a buffer.

## Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to copy in tiles. Range 0 - 31
<i>h</i>	Height of area to copy in tiles. Range 0 - 31
<i>tiles</i>	Pointer to destination buffer for Tile Map data

Entries are copied into **tiles** from the Window Tile Map starting at **x**, **y** reading across for **w** tiles and down for **h** tiles.

One byte per tile.

The buffer pointed to by **tiles** should be at least **x x y** bytes in size.

**19.29.4.48** [set\\_win\\_tile\\_xy\(\)](#) `UINT8* set_win_tile_xy (`  
     [UBYTE](#) *x*,  
     [UBYTE](#) *y*,  
     [UBYTE](#) *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

**19.29.4.49 move\_win()** `void move_win (`  
    `UINT8 x,`  
    `UINT8 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](#)

**19.29.4.50 scroll\_win()** `void scroll_win (`  
    `INT8 x,`  
    `INT8 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> Range: -128 - 127
<i>y</i>	Number of pixels to move the window on the <b>Y axis</b> Range: -128 - 127

**See also**

[move\\_win](#)

**19.29.4.51 set\_sprite\_data()** `void set_sprite_data (`  
    `UINT8 first_tile,`  
    `UINT8 nb_tiles,`  
    `const unsigned char * data )`

Sets VRAM Tile Pattern data for Sprites

**Parameters**

<i>first_tile</i>	Index of the first tile to write
-------------------	----------------------------------

## Parameters

<i>nb_tiles</i>	Number of tiles to write
<i>data</i>	Pointer to (2 bpp) source Tile Pattern data

Writes **nb\_tiles** tiles to VRAM starting at **first\_tile**, tile data is sourced from **data**. Each Tile is 16 bytes in size (8x8 pixels, 2 bits-per-pixel).

Note: Sprite Tiles 128-255 share the same memory region as Background Tiles 128-255.

GBC only: [VBK\\_REG](#) determines which bank of Background tile patterns are written to.

- VBK\_REG=0 indicates the first bank
- VBK\_REG=1 indicates the second

**19.29.4.52 set\_sprite\_1bit\_data()** `void set_sprite_1bit_data (`  
     `UINT8 first_tile,`  
     `UINT8 nb_tiles,`  
     `const unsigned char * 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 color 0
- 1 will be expanded into color 3

## See also

[SHOW\\_SPRITES](#), [HIDE\\_SPRITES](#), [set\\_sprite\\_tile](#)

**19.29.4.53 get\_sprite\_data()** `void get_sprite_data (`  
     `UINT8 first_tile,`  
     `UINT8 nb_tiles,`  
     `unsigned char * 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

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.

**19.29.4.54 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

**19.29.4.55 set\_sprite\_tile()** `void set_sprite_tile (`  
     `UINT8 nb,`  
     `UINT8 tile ) [inline]`

Sets sprite number **nb** in the OAM to display tile number **tile**.

#### Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>tile</i>	Selects a tile (0 - 255) from memory at 8000h - 8FFFh In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the OAM Attribute Flag (see <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](#)

**19.29.4.56 get\_sprite\_tile()** `UINT8 get_sprite_tile (`  
     `UINT8 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

**19.29.4.57 set\_sprite\_prop()** `void set_sprite_prop (`  
     `UINT8 nb,`  
     `UINT8 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.

**19.29.4.58 get\_sprite\_prop()** `UINT8 get_sprite_prop (`  
`UINT8 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

**19.29.4.59 move\_sprite()** `void move_sprite (`  
`UINT8 nb,`  
`UINT8 x,`  
`UINT8 y ) [inline]`

Moves sprite number **nb** to the **x, y** position on the screen.

#### Parameters

<i>nb</i>	Sprite number, range 0 - 39
<i>x</i>	X Position. Specifies the sprites horizontal position on the screen (minus 8). An offscreen value (X=0 or X>=168) hides the sprite, but the sprite still affects the priority ordering - a better way to hide a sprite is to set its Y-coordinate offscreen.
<i>y</i>	Y Position. Specifies the sprites vertical position on the screen (minus 16). An offscreen value (for example, Y=0 or Y>=160) hides the sprite.

Moving the sprite to 0,0 (or similar off-screen location) will hide it.

**19.29.4.60 scroll\_sprite()** `void scroll_sprite (`  
`UINT8 nb,`  
`INT8 x,`  
`INT8 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> Range: -128 - 127
<i>y</i>	Number of pixels to move the sprite on the <b>Y axis</b> Range: -128 - 127

**See also**

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

**19.29.4.61 hide\_sprite()** `void hide_sprite (`  
    `UINT8 nb ) [inline]`

Hides sprite number **nb** by moving it to zero position by Y.

**Parameters**

<i>nb</i>	Sprite number, range 0 - 39
-----------	-----------------------------

**19.29.4.62 set\_data()** `void set_data (`  
    `unsigned char * vram_addr,`  
    `const unsigned char * data,`  
    `UINT16 len )`

Copies Tile Pattern data to an address in VRAM

**Parameters**

<i>vram_addr</i>	Pointer to destination VRAM Address
<i>data</i>	Pointer to source buffer
<i>len</i>	Number of bytes to copy

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

GBC only: [VBK\\_REG](#) determines which bank of Background tile patterns are written to.

- VBK\_REG=0 indicates the first bank
- VBK\_REG=1 indicates the second

**19.29.4.63 get\_data()** `void get_data (`  
    `unsigned char * data,`  
    `unsigned char * vram_addr,`  
    `UINT16 len )`

Copies Tile Pattern data from an address in VRAM into a buffer

**Parameters**

<i>vram_addr</i>	Pointer to source VRAM Address
<i>data</i>	Pointer to destination buffer
<i>len</i>	Number of bytes to copy

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

GBC only: **VBK\_REG** determines which bank of Background tile patterns are written to.

- **VBK\_REG=0** indicates the first bank
- **VBK\_REG=1** indicates the second

**19.29.4.64 set\_tiles()** `void set_tiles (`  
`UINT8 x,`  
`UINT8 y,`  
`UINT8 w,`  
`UINT8 h,`  
`unsigned char * vram_addr,`  
`const unsigned char * 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 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=0** Tile Numbers are written
- **VBK\_REG=1** Tile Attributes are written

**19.29.4.65 get\_tiles()** `void get_tiles (`  
`UINT8 x,`  
`UINT8 y,`  
`UINT8 w,`  
`UINT8 h,`  
`unsigned char * tiles,`  
`unsigned char * vram_addr )`

Copies a rectangular region of Tile Map entries from a given VRAM Address 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
<i>vram_addr</i>	Pointer to source VRAM Address



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.

**19.29.4.66 init\_win()** `void init_win (`  
`UINT8 c )`

Initializes the entire Window Tile Map with Tile Number **c**

#### Parameters

<b>c</b>	Tile number to fill with
----------	--------------------------

Note: This function avoids writes during modes 2 & 3

**19.29.4.67 init\_bkg()** `void init_bkg (`  
`UINT8 c )`

Initializes the entire Background Tile Map with Tile Number **c**

#### Parameters

<b>c</b>	Tile number to fill with
----------	--------------------------

Note: This function avoids writes during modes 2 & 3

**19.29.4.68 vmemset()** `void vmemset (`  
`void * s,`  
`UINT8 c,`  
`size_t n )`

Fills the VRAM memory region **s** of size **n** with Tile Number **c**

#### Parameters

<b>s</b>	Start address in VRAM
<b>c</b>	Tile number to fill with
<b>n</b>	Size of memory region (in bytes) to fill

Note: This function avoids writes during modes 2 & 3

**19.29.4.69 fill\_bkg\_rect()** `void fill_bkg_rect (`  
`UINT8 x,`  
`UINT8 y,`  
`UINT8 w,`  
`UINT8 h,`  
`UINT8 tile )`

Fills a rectangular region of Tile Map entries for the Background layer with tile.

#### Parameters

<b>x</b>	X Start position in Background Map tile coordinates. Range 0 - 31
<b>y</b>	Y Start position in Background Map tile coordinates. Range 0 - 31
<b>w</b>	Width of area to set in tiles. Range 0 - 31
<b>h</b>	Height of area to set in tiles. Range 0 - 31
<b>tile</b>	Fill value

**19.29.4.70 fill\_win\_rect()** `void fill_win_rect (`  
     `UINT8 x,`  
     `UINT8 y,`  
     `UINT8 w,`  
     `UINT8 h,`  
     `UINT8 tile )`

Fills a rectangular region of Tile Map entries for the Window layer with tile.

#### Parameters

<i>x</i>	X Start position in Window Map tile coordinates. Range 0 - 31
<i>y</i>	Y Start position in Window Map tile coordinates. Range 0 - 31
<i>w</i>	Width of area to set in tiles. Range 0 - 31
<i>h</i>	Height of area to set in tiles. Range 0 - 31
<i>tile</i>	Fill value

## 19.29.5 Variable Documentation

**19.29.5.1** `c` `int c`

**19.29.5.2** `_cpu` `UINT8 _cpu`

GB CPU type

See also

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

**19.29.5.3** `sys_time` `volatile UINT16 sys_time`

Global Time Counter in VBL periods (60Hz)

Increments once per Frame

Will wrap around every ~18 minutes (unsigned 16 bits = 65535 / 60 / 60 = 18.2)

**19.29.5.4** `_io_status` `volatile UINT8 _io_status`

Serial Link: Current IO Status. An OR of IO\_\*

**19.29.5.5** `_io_in` `volatile UINT8 _io_in`

Serial Link: Byte just read after calling [receive\\_byte\(\)](#)

**19.29.5.6** `_io_out` `volatile UINT8 _io_out`

Serial Link: Write byte to send here before calling [send\\_byte\(\)](#)

**19.29.5.7** `_current_bank` `__REG _current_bank`

Tracks current active ROM bank

See also

[SWITCH\\_ROM\\_MBC1\(\)](#), [SWITCH\\_ROM\\_MBC5\(\)](#) This variable is updated automatically when you call [SWITCH\\_ROM\\_MBC1](#) or [SWITCH\\_ROM\\_MBC5](#), or call a [BANKED](#) function.

**19.29.5.8** **h** void h

**19.29.5.9** **l** void l

**19.29.5.10** **b** void b

**19.29.5.11** **d** void d

**19.29.5.12** **e** void e

**19.29.5.13** **shadow\_OAM** volatile struct [OAM\\_item\\_t](#) shadow\_OAM[]  
Shadow OAM array in WRAM, that is DMA-transferred into the real OAM each VBlank

**19.29.5.14** **\_shadow\_OAM\_base** [\\_\\_REG](#) \_shadow\_OAM\_base  
MSB of shadow\_OAM address is used by OAM DMA copying routine

## 19.30 gb/gbdecompress.h File Reference

### Functions

- void [gb\\_decompress](#) (const unsigned char \*sour, unsigned char \*dest) \_\_preserves\_regs([b](#))
- void [gb\\_decompress\\_bkg\\_data](#) ([UINT8](#) first\_tile, const unsigned char \*sour) \_\_preserves\_regs([b](#))
- void [gb\\_decompress\\_win\\_data](#) ([UINT8](#) first\_tile, const unsigned char \*sour) \_\_preserves\_regs([b](#))
- void [gb\\_decompress\\_sprite\\_data](#) ([UINT8](#) first\_tile, const unsigned char \*sour) \_\_preserves\_regs([b](#))

### Variables

- void [c](#)

### 19.30.1 Detailed Description

GB-Compress decompressor Compatible with the compression used in GBTD

### 19.30.2 Function Documentation

**19.30.2.1** **gb\_decompress()** void gb\_decompress (  
    const unsigned char \* *sour*,  
    unsigned char \* *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

#### See also

[gb\\_decompress\\_bkg\\_data](#), [gb\\_decompress\\_win\\_data](#), [gb\\_decompress\\_sprite\\_data](#)

**19.30.2.2 gb\_decompress\_bkg\_data()** void gb\_decompress\_bkg\_data (  
    UINT8 first\_tile,  
    const unsigned char \* 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

#### See also

gb\_decompress\_bkg, [gb\\_decompress\\_win\\_data](#), [gb\\_decompress\\_sprite\\_data](#)

**19.30.2.3 gb\_decompress\_win\_data()** void gb\_decompress\_win\_data (  
    UINT8 first\_tile,  
    const unsigned char \* 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

#### See also

[gb\\_decompress](#), [gb\\_decompress\\_bkg\\_data](#), [gb\\_decompress\\_sprite\\_data](#)

**19.30.2.4 gb\_decompress\_sprite\_data()** void gb\_decompress\_sprite\_data (  
    UINT8 first\_tile,  
    const unsigned char \* 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

#### See also

[gb\\_decompress](#), [gb\\_decompress\\_bkg\\_data](#), [gb\\_decompress\\_win\\_data](#)

### 19.30.3 Variable Documentation

**19.30.3.1 c** void c

## 19.31 gb/hardware.h File Reference

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

### Macros

- `#define __REG extern volatile __sfr`

### Variables

- [\\_\\_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](#)
- [\\_\\_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 IE\\_REG](#)

### 19.31.1 Detailed Description

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

### 19.31.2 Macro Definition Documentation

**19.31.2.1** [\\_\\_REG](#) `#define __REG extern volatile __sfr`

### 19.31.3 Variable Documentation

**19.31.3.1** [P1\\_REG](#) [\\_\\_REG](#) `P1_REG`  
Joystick: 1.1.P15.P14.P13.P12.P11.P10

**19.31.3.2** [SB\\_REG](#) [\\_\\_REG](#) `SB_REG`  
Serial IO data buffer

**19.31.3.3** [SC\\_REG](#) [\\_\\_REG](#) `SC_REG`  
Serial IO control register

**19.31.3.4** [DIV\\_REG](#) [\\_\\_REG](#) `DIV_REG`  
Divider register

**19.31.3.5** [TIMA\\_REG](#) [\\_\\_REG](#) `TIMA_REG`  
Timer counter

**19.31.3.6** [TMA\\_REG](#) [\\_\\_REG](#) `TMA_REG`  
Timer modulo

**19.31.3.7** [TAC\\_REG](#) [\\_\\_REG](#) `TAC_REG`  
Timer control

**19.31.3.8** [IF\\_REG](#) [\\_\\_REG](#) `IF_REG`  
Interrupt flags: 0.0.0.JOY.SIO.TIM.LCD.VBL

**19.31.3.9** [NR10\\_REG](#) [\\_\\_REG](#) `NR10_REG`  
Sound Channel 1 Sweep

**19.31.3.10** [NR11\\_REG](#) [\\_\\_REG](#) `NR11_REG`  
Sound Channel 1 Sound length/Wave pattern duty

**19.31.3.11 NR12\_REG** [\\_\\_REG](#) NR12\_REG  
Sound Channel 1 Volume Envelope

**19.31.3.12 NR13\_REG** [\\_\\_REG](#) NR13\_REG  
Sound Channel 1 Frequency Low

**19.31.3.13 NR14\_REG** [\\_\\_REG](#) NR14\_REG  
Sound Channel 1 Frequency High

**19.31.3.14 NR21\_REG** [\\_\\_REG](#) NR21\_REG  
Sound Channel 2 Tone

**19.31.3.15 NR22\_REG** [\\_\\_REG](#) NR22\_REG  
Sound Channel 2 Volume Envelope

**19.31.3.16 NR23\_REG** [\\_\\_REG](#) NR23\_REG  
Sound Channel 2 Frequency data Low

**19.31.3.17 NR24\_REG** [\\_\\_REG](#) NR24\_REG  
Sound Channel 2 Frequency data High

**19.31.3.18 NR30\_REG** [\\_\\_REG](#) NR30\_REG  
Sound Channel 3 Sound on/off

**19.31.3.19 NR31\_REG** [\\_\\_REG](#) NR31\_REG  
Sound Channel 3 Sound Length

**19.31.3.20 NR32\_REG** [\\_\\_REG](#) NR32\_REG  
Sound Channel 3 Select output level

**19.31.3.21 NR33\_REG** [\\_\\_REG](#) NR33\_REG  
Sound Channel 3 Frequency data Low

**19.31.3.22 NR34\_REG** [\\_\\_REG](#) NR34\_REG  
Sound Channel 3 Frequency data High

**19.31.3.23 NR41\_REG** [\\_\\_REG](#) NR41\_REG  
Sound Channel 4 Sound Length

**19.31.3.24 NR42\_REG** [\\_\\_REG](#) NR42\_REG  
Sound Channel 4 Volume Envelope

**19.31.3.25 NR43\_REG** [\\_\\_REG](#) NR43\_REG  
Sound Channel 4 Polynomial Counter

**19.31.3.26 NR44\_REG** [\\_\\_REG](#) NR44\_REG  
Sound Channel 4 Counter / Consecutive and Initial

**19.31.3.27 NR50\_REG** [\\_\\_REG](#) NR50\_REG  
Sound Channel control / ON-OFF / Volume

**19.31.3.28 NR51\_REG** [\\_\\_REG](#) NR51\_REG  
Sound Selection of Sound output terminal

**19.31.3.29 NR52\_REG** [\\_\\_REG](#) NR52\_REG  
Sound Master on/off

**19.31.3.30 LCDC\_REG** [\\_\\_REG](#) LCDC\_REG  
LCD control

**19.31.3.31 STAT\_REG** [\\_\\_REG](#) STAT\_REG  
LCD status

**19.31.3.32 SCY\_REG** [\\_\\_REG](#) SCY\_REG  
Scroll Y

**19.31.3.33 SCX\_REG** [\\_\\_REG](#) SCX\_REG  
Scroll X

**19.31.3.34 LY\_REG** [\\_\\_REG](#) LY\_REG  
LCDC Y-coordinate

**19.31.3.35 LYC\_REG** [\\_\\_REG](#) LYC\_REG  
LY compare

**19.31.3.36 DMA\_REG** [\\_\\_REG](#) DMA\_REG  
DMA transfer

**19.31.3.37 BGP\_REG** [\\_\\_REG](#) BGP\_REG  
BG palette data

**19.31.3.38 OBP0\_REG** [\\_\\_REG](#) OBP0\_REG  
OBJ palette 0 data

**19.31.3.39 OBP1\_REG** [\\_\\_REG](#) OBP1\_REG  
OBJ palette 1 data

**19.31.3.40 WY\_REG** [\\_\\_REG](#) WY\_REG  
Window Y coordinate

**19.31.3.41 WX\_REG** [\\_\\_REG](#) WX\_REG  
Window X coordinate

**19.31.3.42 KEY1\_REG** [\\_\\_REG](#) KEY1\_REG  
CPU speed

**19.31.3.43 VBK\_REG** [\\_\\_REG](#) VBK\_REG  
VRAM bank

**19.31.3.44 HDMA1\_REG** [\\_\\_REG](#) HDMA1\_REG  
DMA control 1

**19.31.3.45 HDMA2\_REG** [\\_\\_REG](#) HDMA2\_REG  
DMA control 2

**19.31.3.46 HDMA3\_REG** [\\_\\_REG](#) HDMA3\_REG  
DMA control 3



**19.31.3.47 HDMA4\_REG** [\\_\\_REG](#) HDMA4\_REG  
DMA control 4

**19.31.3.48 HDMA5\_REG** [\\_\\_REG](#) HDMA5\_REG  
DMA control 5

**19.31.3.49 RP\_REG** [\\_\\_REG](#) RP\_REG  
IR port

**19.31.3.50 BCPS\_REG** [\\_\\_REG](#) BCPS\_REG  
BG color palette specification

**19.31.3.51 BCPD\_REG** [\\_\\_REG](#) BCPD\_REG  
BG color palette data

**19.31.3.52 OCPS\_REG** [\\_\\_REG](#) OCPS\_REG  
OBJ color palette specification

**19.31.3.53 OCPD\_REG** [\\_\\_REG](#) OCPD\_REG  
OBJ color palette data

**19.31.3.54 SVBK\_REG** [\\_\\_REG](#) SVBK\_REG  
WRAM bank

**19.31.3.55 IE\_REG** [\\_\\_REG](#) IE\_REG  
Interrupt enable

## 19.32 gb/malloc.h File Reference

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

### Data Structures

- struct [smalloc\\_hunk](#)

### Macros

- #define [MALLOC\\_FREE](#) 1
- #define [MALLOC\\_USED](#) 2
- #define [MALLOC\\_MAGIC](#) 123

### Typedefs

- typedef struct [smalloc\\_hunk](#) [mmalloc\\_hunk](#)
- typedef struct [smalloc\\_hunk](#) \* [pmmalloc\\_hunk](#)

### Functions

- void [malloc\\_gc](#) (void) [NONBANKED](#)
- void [debug](#) (char \*routine, char \*msg) [NONBANKED](#)

### Variables

- [UBYTE](#) [malloc\\_heap\\_start](#)
- [pmmalloc\\_hunk](#) [malloc\\_first](#)

### 19.32.1 Detailed Description

Header for a simple implementation of [malloc\(\)](#).

**Todo** : This library may currently be broken.

### 19.32.2 Macro Definition Documentation

#### 19.32.2.1 MALLOC\_FREE `#define MALLOC_FREE 1`

The malloc hunk flags Note: Could have used a negative size a'la TI

#### 19.32.2.2 MALLOC\_USED `#define MALLOC_USED 2`

#### 19.32.2.3 MALLOC\_MAGIC `#define MALLOC_MAGIC 123`

Magic number of a header. Gives us some chance of surviving if the list is corrupted

### 19.32.3 Typedef Documentation

#### 19.32.3.1 mmalloc\_hunk `typedef struct smalloc_hunk mmalloc_hunk`

#### 19.32.3.2 pmmalloc\_hunk `typedef struct smalloc_hunk* pmmalloc_hunk`

### 19.32.4 Function Documentation

#### 19.32.4.1 malloc\_gc() `void malloc_gc ( void )`

Garbage collect (join free hunks)

#### 19.32.4.2 debug() `void debug ( char * routine, char * msg )`

debug message logger

### 19.32.5 Variable Documentation

#### 19.32.5.1 malloc\_heap\_start `UBYTE malloc_heap_start`

Start of free memory, as defined by the linker

#### 19.32.5.2 malloc\_first `pmmalloc_hunk malloc_first`

First hunk

## 19.33 gb/metaspprites.h File Reference

### Data Structures

- struct [metasprite\\_t](#)

## Macros

- `#define metasprite_end -128`

## Typedefs

- `typedef struct metasprite_t metasprite_t`

## Functions

- `UBYTE move_metasprite` (const `metasprite_t` \*metasprite, `UINT8` base\_tile, `UINT8` base\_sprite, `UINT8` x, `UINT8` y)
- `UBYTE move_metasprite_vflip` (const `metasprite_t` \*metasprite, `UINT8` base\_tile, `UINT8` base\_sprite, `UINT8` x, `UINT8` y)
- `UBYTE move_metasprite_hflip` (const `metasprite_t` \*metasprite, `UINT8` base\_tile, `UINT8` base\_sprite, `UINT8` x, `UINT8` y)
- `UBYTE move_metasprite_hvflip` (const `metasprite_t` \*metasprite, `UINT8` base\_tile, `UINT8` base\_sprite, `UINT8` x, `UINT8` y)
- `void hide_metasprite` (const `metasprite_t` \*metasprite, `UINT8` base\_sprite)

## Variables

- `const void * __current_metasprite`
- `UBYTE __current_base_tile`
- `UBYTE __render_shadow_OAM`

### 19.33.1 Detailed Description

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_png2mtspr` tool to convert single or multiple frames of graphics into metasprite structured data for use with the `...metasprite...()` functions.

### 19.33.2 Macro Definition Documentation

#### 19.33.2.1 `metasprite_end` `#define metasprite_end -128`

### 19.33.3 Typedef Documentation

#### 19.33.3.1 `metasprite_t` `typedef struct metasprite_t metasprite_t`

Metasprite sub-item structure

#### Parameters

<i>dy</i>	(INT8) Y coordinate of the sprite relative to the metasprite origin (pivot)
<i>dx</i>	(INT8) X coordinate of the sprite relative to the metasprite origin (pivot)
<i>dtile</i>	(UINT8) Start tile relative to the metasprites own set of tiles
<i>props</i>	(UINT8) Property Flags

Metaspprites 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).

### 19.33.4 Function Documentation

**19.33.4.1 move\_metasprite()** `UBYTE move_metasprite (`  
`const metasprite_t * metasprite,`  
`UINT8 base_tile,`  
`UINT8 base_sprite,`  
`UINT8 x,`  
`UINT8 y ) [inline]`

Moves metasprite to the absolute position x and y

#### Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Moves **metasprite** to the absolute position **x** and **y** (with **no flip** on the X or Y axis). Hardware sprites are allocated starting from **base\_sprite**, using tiles starting from **base\_tile**.

Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

#### Returns

Number of hardware sprites used to draw this metasprite

**19.33.4.2 move\_metasprite\_vflip()** `UBYTE move_metasprite_vflip (`  
`const metasprite_t * metasprite,`  
`UINT8 base_tile,`  
`UINT8 base_sprite,`  
`UINT8 x,`  
`UINT8 y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the Y axis**

#### Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move\\_metasprite\(\)](#), but with the metasprite flipped on the Y axis only.

Sets:

- `__current metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Returns

Number of hardware sprites used to draw this metasprite

See also

[move\\_metasprite\(\)](#)

**19.33.4.3 move\_metasprite\_hflip()** `UBYTE move_metasprite_hflip (`  
    `const metasprite_t * metasprite,`  
    `UINT8 base_tile,`  
    `UINT8 base_sprite,`  
    `UINT8 x,`  
    `UINT8 y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X axis**

Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move\\_metasprite\(\)](#), but with the metasprite flipped on the X axis only.

Sets:

- `__current metasprite = metasprite;`
- `__current_base_tile = base_tile;`

Returns

Number of hardware sprites used to draw this metasprite

See also

[move\\_metasprite\(\)](#)

**19.33.4.4 move\_metasprite\_hvflip()** `UBYTE move_metasprite_hvflip (`  
    `const metasprite_t * metasprite,`  
    `UINT8 base_tile,`  
    `UINT8 base_sprite,`  
    `UINT8 x,`  
    `UINT8 y ) [inline]`

Moves metasprite to the absolute position x and y, **flipped on the X and Y axis**

## Parameters

<i>metasprite</i>	Pointer to the first struct of the metasprite (for the desired frame)
<i>base_tile</i>	Number of the first tile where the metasprite's tiles start
<i>base_sprite</i>	Number of the first hardware sprite to be used by the metasprite
<i>x</i>	Absolute x coordinate of the sprite
<i>y</i>	Absolute y coordinate of the sprite

Same as [move\\_metasprite\(\)](#), but with the metasprite flipped on both the X and Y axis.  
Sets:

- `__current_metasprite = metasprite;`
- `__current_base_tile = base_tile;`

## Returns

Number of hardware sprites used to draw this metasprite

## See also

[move\\_metasprite\(\)](#)

**19.33.4.5 hide\_metasprite()** `void hide_metasprite (`  
`const metasprite_t * metasprite,`  
`UINT8 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;`

**19.33.5 Variable Documentation**

**19.33.5.1 \_\_current\_metasprite** `const void* __current_metasprite`

**19.33.5.2 \_\_current\_base\_tile** `UBYTE __current_base_tile`

**19.33.5.3 \_\_render\_shadow\_OAM** `UBYTE __render_shadow_OAM`

**19.34 gb/sample.h File Reference**

## Functions

- void [play\\_sample](#) (UINT8 \*start, UINT16 len) **NONBANKED**

### 19.34.1 Detailed Description

Playback raw sound sample with length `len` from start at 8192Hz rate. `len` defines the length of the sample in samples/32 or bytes/16. The format of the data is unsigned 4-bit samples, 2 samples per byte, upper 4-bits played before lower 4 bits.

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

### 19.34.2 Function Documentation

**19.34.2.1 `play_sample()`** `void play_sample (`  
    `UINT8 * start,`  
    `UINT16 len )`

Play the given, appropriately formatted sample.

## 19.35 gb/sgb.h File Reference

### 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 sgb_check (void)`
- `void sgb_transfer (unsigned char *packet)`
- `void sgb_transfer_nowait (unsigned char *packet)`

### 19.35.1 Detailed Description

Super Gameboy definitions.

See the example SGB project for additional details.

## 19.35.2 Macro Definition Documentation

**19.35.2.1 SGB\_PAL\_01** `#define SGB_PAL_01 0x00U`

SGB Command: Set SGB Palettes 0 & 1

**19.35.2.2 SGB\_PAL\_23** `#define SGB_PAL_23 0x01U`

SGB Command: Set SGB Palettes 2 & 3

**19.35.2.3 SGB\_PAL\_03** `#define SGB_PAL_03 0x02U`

SGB Command: Set SGB Palettes 0 & 3

**19.35.2.4 SGB\_PAL\_12** `#define SGB_PAL_12 0x03U`

SGB Command: Set SGB Palettes 1 & 2

**19.35.2.5 SGB\_ATTR\_BLK** `#define SGB_ATTR_BLK 0x04U`

SGB Command: Set color attributes for rectangular regions

**19.35.2.6 SGB\_ATTR\_LIN** `#define SGB_ATTR_LIN 0x05U`

SGB Command: Set color attributes for horizontal or vertical character lines

**19.35.2.7 SGB\_ATTR\_DIV** `#define SGB_ATTR_DIV 0x06U`

SGB Command: Split screen in half and assign separate color attributes to each side and the divider

**19.35.2.8 SGB\_ATTR\_CHR** `#define SGB_ATTR_CHR 0x07U`

SGB Command: Set color attributes for separate charactersSet SGB Palette 0,1 Data

**19.35.2.9 SGB\_SOUND** `#define SGB_SOUND 0x08U`

SGB Command: Start and stop a internal sound effect, and sounds using internal tone data

**19.35.2.10 SGB\_SOU\_TRN** `#define SGB_SOU_TRN 0x09U`

SGB Command: Transfer sound code or data to the SNES APU RAM

**19.35.2.11 SGB\_PAL\_SET** `#define SGB_PAL_SET 0x0AU`

SGB Command: Apply (previously transferred) SGB system color palettes to actual SNES palettes

**19.35.2.12 SGB\_PAL\_TRN** `#define SGB_PAL_TRN 0x0BU`

SGB Command: Transfer palette data into SGB system color palettes

**19.35.2.13 SGB\_ATTRC\_EN** `#define SGB_ATTRC_EN 0x0CU`

SGB Command: Enable/disable Attraction mode. It is enabled by default

**19.35.2.14 SGB\_TEST\_EN** `#define SGB_TEST_EN 0x0DU`

SGB Command: Enable/disable test mode for "SGB-CPU variable clock speed function"

**19.35.2.15 SGB\_ICON\_EN** `#define SGB_ICON_EN 0x0EU`

SGB Command: Enable/disable ICON functionality

**19.35.2.16 SGB\_DATA\_SND** `#define SGB_DATA_SND 0x0FU`

SGB Command: Write one or more bytes into SNES Work RAM

**19.35.2.17 SGB\_DATA\_TRN** `#define SGB_DATA_TRN 0x10U`

SGB Command: Transfer code or data into SNES RAM



**19.35.2.18 SGB\_MLT\_REQ** `#define SGB_MLT_REQ 0x11U`

SGB Command: Request multiplayer mode (input from more than one joypad)

**19.35.2.19 SGB\_JUMP** `#define SGB_JUMP 0x12U`

SGB Command: Set the SNES program counter and NMI (vblank interrupt) handler to specific addresses

**19.35.2.20 SGB\_CHR\_TRN** `#define SGB_CHR_TRN 0x13U`

SGB Command: Transfer tile data (characters) to SNES Tile memory

**19.35.2.21 SGB\_PCT\_TRN** `#define SGB_PCT_TRN 0x14U`

SGB Command: Transfer tile map and palette data to SNES BG Map memory

**19.35.2.22 SGB\_ATTR\_TRN** `#define SGB_ATTR_TRN 0x15U`

SGB Command: Transfer data to (color) Attribute Files (ATFs) in SNES RAM

**19.35.2.23 SGB\_ATTR\_SET** `#define SGB_ATTR_SET 0x16U`

SGB Command: Transfer attributes from (color) Attribute Files (ATF) to the Game Boy window

**19.35.2.24 SGB\_MASK\_EN** `#define SGB_MASK_EN 0x17U`

SGB Command: Modify Game Boy window mask settings

**19.35.2.25 SGB\_OBJ\_TRN** `#define SGB_OBJ_TRN 0x18U`

SGB Command: Transfer OBJ attributes to SNES OAM memory

**19.35.3 Function Documentation****19.35.3.1 `sgb_check()`** `UINT8 sgb_check (void)`

Returns a non-null value if running on Super GameBoy

**19.35.3.2 `sgb_transfer()`** `void sgb_transfer (unsigned char * packet)`

Transfer a SGB packet

**Parameters**

<code>packet</code>	Pointer to buffer with SGB packet data.
---------------------	---

The first byte of **packet** should be a SGB command, then up to 15 bytes of command parameter data. See the `sgb_border` GBDK example project for a demo of how to use these the `sgb` functions.

See also

[`sgb\_check\(\)`](#)

**19.35.3.3 `sgb_transfer_nowait()`** `void sgb_transfer_nowait (unsigned char * packet)`

Transfer a SGB packet without the 60 ms / 4 frame delay at the end (the delay time is required between consecutive SGB packets)

**Parameters**

<code>packet</code>	Pointer to buffer with SGB packet data.
---------------------	---

See also

[sgb\\_transfer\(\)](#)

## 19.36 gbdk-lib.h File Reference

```
#include <asm/gbz80/provides.h>
```

### 19.36.1 Detailed Description

Settings for the greater library system.

## 19.37 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

### 19.37.1 Macro Definition Documentation

**19.37.1.1 CHAR\_BIT** #define CHAR\_BIT 8 /\* bits in a char \*/

**19.37.1.2 SCHAR\_MAX** #define SCHAR\_MAX 127

**19.37.1.3 SCHAR\_MIN** #define SCHAR\_MIN -128

**19.37.1.4 UCHAR\_MAX** #define UCHAR\_MAX 0xff

**19.37.1.5 CHAR\_MAX** #define CHAR\_MAX [SCHAR\\_MAX](#)

**19.37.1.6 CHAR\_MIN** #define CHAR\_MIN [SCHAR\\_MIN](#)

**19.37.1.7 INT\_MIN** `#define INT_MIN (-32767 - 1)`

**19.37.1.8 INT\_MAX** `#define INT_MAX 32767`

**19.37.1.9 SHRT\_MAX** `#define SHRT_MAX INT_MAX`

**19.37.1.10 SHRT\_MIN** `#define SHRT_MIN INT_MIN`

**19.37.1.11 UINT\_MAX** `#define UINT_MAX 0xffff`

**19.37.1.12 UINT\_MIN** `#define UINT_MIN 0`

**19.37.1.13 USHRT\_MAX** `#define USHRT_MAX UINT_MAX`

**19.37.1.14 USHRT\_MIN** `#define USHRT_MIN UINT_MIN`

**19.37.1.15 LONG\_MIN** `#define LONG_MIN (-2147483647L-1)`

**19.37.1.16 LONG\_MAX** `#define LONG_MAX 2147483647L`

**19.37.1.17 ULONG\_MAX** `#define ULONG_MAX 0xffffffff`

**19.37.1.18 ULONG\_MIN** `#define ULONG_MIN 0`

## 19.38 rand.h File Reference

`#include <types.h>`

### Functions

- void `initrand` (`UINT16` seed) `NONBANKED`
- `INT8` `rand` (void)
- `UINT16` `randw` (void)
- void `initarand` (`UINT16` seed)
- `INT8` `arand` (void)

### 19.38.1 Detailed Description

Random generator using the linear congruential method

Author

Luc Van den Borre

## 19.38.2 Function Documentation

**19.38.2.1 `initrand()`** `void initrand (`  
`UINT16 seed )`

Initialise the pseudo-random number generator.

### Parameters

<code>seed</code>	The value for initializing the random number generator.
-------------------	---

The seed should be different each time, otherwise the same pseudo-random sequence will be generated.

The DIV Register ([DIV\\_REG](#)) is sometimes used as a seed, particularly if read at some variable point in time (such as when the player presses a button).

Only needs to be called once to initialize, but may be called again to re-initialize with the same or a different seed.

See also

[rand\(\)](#), [randw\(\)](#)

**19.38.2.2 `rand()`** `INT8 rand (`  
`void )`

Returns a random byte (8 bit) value.

[initrand\(\)](#) should be used to initialize the random number generator before using [rand\(\)](#)

**19.38.2.3 `randw()`** `UINT16 randw (`  
`void )`

Returns a random word (16 bit) value.

[initrand\(\)](#) should be used to initialize the random number generator before using [rand\(\)](#)

**19.38.2.4 `initarand()`** `void initarand (`  
`UINT16 seed )`

Random generator using the linear lagged additive method

### Parameters

<code>seed</code>	The value for initializing the random number generator.
-------------------	---

Note: [initarand\(\)](#) calls [initrand\(\)](#) with the same seed value, and uses [rand\(\)](#) to initialize the random generator.

See also

[initrand\(\)](#) for suggestions about seed values, [arand\(\)](#)

**19.38.2.5 `arand()`** `INT8 arand (`  
`void )`

Returns a random number generated with the linear lagged additive method.

[initarand\(\)](#) should be used to initialize the random number generator before using [arand\(\)](#)

## 19.39 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)`
- `_Noreturn void longjmp (jmp_buf, int)`

## 19.39.1 Macro Definition Documentation

**19.39.1.1 SP\_SIZE** `#define SP_SIZE 1`

**19.39.1.2 BP\_SIZE** `#define BP_SIZE 0`

**19.39.1.3 SPX\_SIZE** `#define SPX_SIZE 0`

**19.39.1.4 BPX\_SIZE** `#define BPX_SIZE SPX_SIZE`

**19.39.1.5 RET\_SIZE** `#define RET_SIZE 2`

**19.39.1.6 setjmp** `#define setjmp(  
jump_buf ) __setjmp(jump_buf)`

## 19.39.2 Typedef Documentation

**19.39.2.1 jmp\_buf** `typedef unsigned char jmp_buf[RET_SIZE+SP_SIZE+BP_SIZE+SPX_SIZE+BPX_SIZE]`

## 19.39.3 Function Documentation

**19.39.3.1 \_\_setjmp()** `int __setjmp (  
jmp_buf )`

**19.39.3.2 longjmp()** `_Noreturn void longjmp (  
jmp_buf ,  
int )`

## 19.40 stdatomic.h File Reference

### Data Structures

- struct [atomic\\_flag](#)

### Functions

- `_Bool` [atomic\\_flag\\_test\\_and\\_set](#) (volatile [atomic\\_flag](#) \*object)
- void [atomic\\_flag\\_clear](#) (volatile [atomic\\_flag](#) \*object)

#### 19.40.1 Function Documentation

**19.40.1.1 [atomic\\_flag\\_test\\_and\\_set\(\)](#)** `_Bool atomic_flag_test_and_set ( volatile atomic\_flag * object )`

**19.40.1.2 [atomic\\_flag\\_clear\(\)](#)** `void atomic_flag_clear ( volatile atomic\_flag * object )`

## 19.41 stdbool.h File Reference

### Macros

- `#define` [true](#) `((_Bool)+1)`
- `#define` [false](#) `((_Bool)+0)`
- `#define` [bool](#) `_Bool`
- `#define` [\\_\\_bool\\_true\\_false\\_are\\_defined](#) `1`

#### 19.41.1 Macro Definition Documentation

**19.41.1.1 [true](#)** `#define true ((_Bool)+1)`

**19.41.1.2 [false](#)** `#define false ((_Bool)+0)`

**19.41.1.3 [bool](#)** `#define bool _Bool`

**19.41.1.4 [\\_\\_bool\\_true\\_false\\_are\\_defined](#)** `#define __bool_true_false_are_defined 1`

## 19.42 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](#)

### 19.42.1 Macro Definition Documentation

**19.42.1.1 NULL** `#define NULL (void *)0`

**19.42.1.2 \_\_PTRDIFF\_T\_DEFINED** `#define __PTRDIFF_T_DEFINED`

**19.42.1.3 \_\_SIZE\_T\_DEFINED** `#define __SIZE_T_DEFINED`

**19.42.1.4 \_\_WCHAR\_T\_DEFINED** `#define __WCHAR_T_DEFINED`

**19.42.1.5 offsetof** `#define offsetof(  
    s,  
    m ) __builtin_offsetof (s, m)`

### 19.42.2 Typedef Documentation

**19.42.2.1 ptrdiff\_t** typedef int [ptrdiff\\_t](#)

**19.42.2.2 size\_t** typedef unsigned int [size\\_t](#)

**19.42.2.3 wchar\_t** typedef unsigned long int [wchar\\_t](#)

## 19.43 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](#)

### 19.43.1 Macro Definition Documentation

**19.43.1.1 INT8\_MIN** `#define INT8_MIN (-128)`

**19.43.1.2 INT16\_MIN** `#define INT16_MIN (-32767-1)`

**19.43.1.3 INT32\_MIN** `#define INT32_MIN (-2147483647L-1)`

**19.43.1.4 INT8\_MAX** `#define INT8_MAX (127)`

**19.43.1.5 INT16\_MAX** `#define INT16_MAX (32767)`

**19.43.1.6 INT32\_MAX** `#define INT32_MAX (2147483647L)`

**19.43.1.7 UINT8\_MAX** `#define UINT8_MAX (255)`

**19.43.1.8 UINT16\_MAX** `#define UINT16_MAX (65535)`

**19.43.1.9 UINT32\_MAX** `#define UINT32_MAX (4294967295UL)`

**19.43.1.10 INT\_LEAST8\_MIN** `#define INT_LEAST8_MIN INT8\_MIN`

**19.43.1.11 INT\_LEAST16\_MIN** `#define INT_LEAST16_MIN INT16\_MIN`

**19.43.1.12 INT\_LEAST32\_MIN** `#define INT_LEAST32_MIN INT32\_MIN`

**19.43.1.13 INT\_LEAST8\_MAX** `#define INT_LEAST8_MAX INT8\_MAX`

**19.43.1.14 INT\_LEAST16\_MAX** `#define INT_LEAST16_MAX INT16\_MAX`

**19.43.1.15 INT\_LEAST32\_MAX** `#define INT_LEAST32_MAX INT32\_MAX`

**19.43.1.16** **UINT\_LEAST8\_MAX** `#define` `UINT_LEAST8_MAX` `UINT8_MAX`

**19.43.1.17** **UINT\_LEAST16\_MAX** `#define` `UINT_LEAST16_MAX` `UINT16_MAX`

**19.43.1.18** **UINT\_LEAST32\_MAX** `#define` `UINT_LEAST32_MAX` `UINT32_MAX`

**19.43.1.19** **INT\_FAST8\_MIN** `#define` `INT_FAST8_MIN` `INT8_MIN`

**19.43.1.20** **INT\_FAST16\_MIN** `#define` `INT_FAST16_MIN` `INT16_MIN`

**19.43.1.21** **INT\_FAST32\_MIN** `#define` `INT_FAST32_MIN` `INT32_MIN`

**19.43.1.22** **INT\_FAST8\_MAX** `#define` `INT_FAST8_MAX` `INT8_MAX`

**19.43.1.23** **INT\_FAST16\_MAX** `#define` `INT_FAST16_MAX` `INT16_MAX`

**19.43.1.24** **INT\_FAST32\_MAX** `#define` `INT_FAST32_MAX` `INT32_MAX`

**19.43.1.25** **UINT\_FAST8\_MAX** `#define` `UINT_FAST8_MAX` `UINT8_MAX`

**19.43.1.26** **UINT\_FAST16\_MAX** `#define` `UINT_FAST16_MAX` `UINT16_MAX`

**19.43.1.27** **UINT\_FAST32\_MAX** `#define` `UINT_FAST32_MAX` `UINT32_MAX`

**19.43.1.28** **INTPTR\_MIN** `#define` `INTPTR_MIN` `(-32767-1)`

**19.43.1.29** **INTPTR\_MAX** `#define` `INTPTR_MAX` `(32767)`

**19.43.1.30** **UINTPTR\_MAX** `#define` `UINTPTR_MAX` `(65535)`

**19.43.1.31** **INTMAX\_MIN** `#define` `INTMAX_MIN` `(-2147483647L-1)`

**19.43.1.32** **INTMAX\_MAX** `#define` `INTMAX_MAX` `(2147483647L)`

**19.43.1.33** **UINTMAX\_MAX** `#define` `UINTMAX_MAX` `(4294967295UL)`

**19.43.1.34 PTRDIFF\_MIN** `#define PTRDIFF_MIN (-32767-1)`

**19.43.1.35 PTRDIFF\_MAX** `#define PTRDIFF_MAX (32767)`

**19.43.1.36 SIG\_ATOMIC\_MIN** `#define SIG_ATOMIC_MIN (0)`

**19.43.1.37 SIG\_ATOMIC\_MAX** `#define SIG_ATOMIC_MAX (255)`

**19.43.1.38 SIZE\_MAX** `#define SIZE_MAX (65535u)`

**19.43.1.39 INT8\_C** `#define INT8_C(  
    c ) c`

**19.43.1.40 INT16\_C** `#define INT16_C(  
    c ) c`

**19.43.1.41 INT32\_C** `#define INT32_C(  
    c ) c ## L`

**19.43.1.42 UINT8\_C** `#define UINT8_C(  
    c ) c ## U`

**19.43.1.43 UINT16\_C** `#define UINT16_C(  
    c ) c ## U`

**19.43.1.44 UINT32\_C** `#define UINT32_C(  
    c ) c ## UL`

**19.43.1.45 WCHAR\_MIN** `#define WCHAR_MIN 0`

**19.43.1.46 WCHAR\_MAX** `#define WCHAR_MAX 0xffffffff`

**19.43.1.47 WINT\_MIN** `#define WINT_MIN 0`

**19.43.1.48 WINT\_MAX** `#define WINT_MAX 0xffffffff`

**19.43.1.49 INTMAX\_C** `#define INTMAX_C(  
    c ) c ## L`

**19.43.1.50 UINTMAX\_C** `#define UINTMAX_C(  
    c ) c ## UL`

## 19.43.2 Typedef Documentation

**19.43.2.1 int8\_t** `typedef signed char int8_t`

**19.43.2.2 int16\_t** `typedef short int int16_t`

**19.43.2.3 int32\_t** `typedef long int int32_t`

**19.43.2.4 uint8\_t** `typedef unsigned char uint8_t`

**19.43.2.5 uint16\_t** `typedef unsigned short int uint16_t`

**19.43.2.6 uint32\_t** `typedef unsigned long int uint32_t`

**19.43.2.7 int\_least8\_t** `typedef signed char int_least8_t`

**19.43.2.8 int\_least16\_t** `typedef short int int_least16_t`

**19.43.2.9 int\_least32\_t** `typedef long int int_least32_t`

**19.43.2.10 uint\_least8\_t** `typedef unsigned char uint_least8_t`

**19.43.2.11 uint\_least16\_t** `typedef unsigned short int uint_least16_t`

**19.43.2.12 uint\_least32\_t** `typedef unsigned long int uint_least32_t`

**19.43.2.13 int\_fast8\_t** `typedef signed char int_fast8_t`

**19.43.2.14 int\_fast16\_t** `typedef int int_fast16_t`

**19.43.2.15 int\_fast32\_t** `typedef long int int_fast32_t`

**19.43.2.16 uint\_fast8\_t** `typedef unsigned char uint_fast8_t`

**19.43.2.17** `uint_fast16_t` typedef unsigned int `uint_fast16_t`

**19.43.2.18** `uint_fast32_t` typedef unsigned long int `uint_fast32_t`

**19.43.2.19** `intptr_t` typedef int `intptr_t`

**19.43.2.20** `uintptr_t` typedef unsigned int `uintptr_t`

**19.43.2.21** `intmax_t` typedef long int `intmax_t`

**19.43.2.22** `uintmax_t` typedef unsigned long int `uintmax_t`

## 19.44 stdio.h File Reference

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

### Functions

- void `putchar` (char `c`)
- void `printf` (const char \*`format`,...) **NONBANKED**
- void `sprintf` (char \*`str`, const char \*`format`,...) **NONBANKED**
- void `puts` (const char \*`s`) **NONBANKED**
- char \* `gets` (char \*`s`)
- char `getchar` (void)

### 19.44.1 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.

### 19.44.2 Function Documentation

**19.44.2.1** `putchar()` void `putchar` (  
    char `c` )

Write the character `c` to stdout.

**19.44.2.2** `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 <code>printf</code>
---------------	--

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.

**19.44.2.3 `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.

**19.44.2.4 `puts()`** `void puts (`  
`const char * s )`

[puts\(\)](#) writes the string **s** and a trailing newline to stdout.

**19.44.2.5 `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**

**19.44.2.6 `getchar()`** `char getchar (`  
`void )`

[getchar\(\)](#) Reads and returns a single character from stdin.

## 19.45 `stdlib.h` File Reference

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

## Macros

- `#define __reentrant`

## Functions

- void `exit` (int status) **NONBANKED**
- int `abs` (int i)
- long `labs` (long num)
- int `atoi` (const char \*s)
- long `atol` (const char \*s)
- char \* `itoa` (int n, char \*s)
- char \* `utoa` (unsigned int n, char \*s)
- char \* `ltoa` (long n, char \*s)
- char \* `ultoa` (unsigned long n, char \*s)
- 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`

### 19.45.1 Macro Definition Documentation

**19.45.1.1 `__reentrant`** `#define __reentrant`  
file `stdlib.h` 'Standard library' functions, for whatever that means.

### 19.45.2 Function Documentation

**19.45.2.1 `exit()`** `void exit (`  
`int status )`

Causes normal program termination and the value of status is returned to the parent. All open streams are flushed and closed.

**19.45.2.2 `abs()`** `int abs (`  
`int i )`

Returns the absolute value of int **i**

#### Parameters

<i>i</i>	Int to obtain absolute value of
----------	---------------------------------

If **i** is negative, returns -i; else returns **i**.

**19.45.2.3 `labs()`** `long labs (`  
`long num )`

Returns the absolute value of long int **num**

#### Parameters

<i>num</i>	Long integer to obtain absolute value of
------------	--

**19.45.2.4 `atoi()`** `int atoi (`  
    `const char * s )`

Converts an ASCII string to an int

**Parameters**

<code>s</code>	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

**19.45.2.5 `atol()`** `long atol (`  
    `const char * s )`

Converts an ASCII string to a long.

**Parameters**

<code>s</code>	String to convert to an long int
----------------	----------------------------------

**See also**

[atoi\(\)](#)

Returns: Long int value of string

**19.45.2.6 `itoa()`** `char* itoa (`  
    `int n,`  
    `char * s )`

Converts an int into a base 10 ASCII string.

**Parameters**

<code>n</code>	Int to convert to a string
<code>s</code>	String to store the converted number

Returns: Pointer to converted string

**19.45.2.7 `utoa()`** `char* utoa (`  
    `unsigned int n,`  
    `char * s )`

Converts an unsigned int into a base 10 ASCII string.

**Parameters**

<code>n</code>	Unsigned Int to convert to a string
<code>s</code>	String to store the converted number

Returns: Pointer to converted string

**19.45.2.8 `ltoa()`** `char* ltoa (`  
    `long n,`



```
char * s )
```

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

Returns: Pointer to converted string

**19.45.2.9 ultoa()** `char* ultoa (`  
     `unsigned long n,`  
     `char * s )`

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

Returns: Pointer to converted string

**19.45.2.10 calloc()** `void* calloc (`  
     `size_t nmemb,`  
     `size_t size )`

Memory allocation functions

**19.45.2.11 malloc()** `void* malloc (`  
     `size_t size )`

**19.45.2.12 realloc()** `void* realloc (`  
     `void * ptr,`  
     `size_t size )`

**19.45.2.13 free()** `void free (`  
     `void * ptr )`

**19.45.2.14 bsearch()** `void* bsearch (`  
     `const void * key,`  
     `const void * base,`  
     `size_t nmemb,`  
     `size_t size,`  
     `int (*)(const void *, const void *) __reentrant compar )`

search a sorted array of **nmemb** items

#### Parameters

<i>key</i>	Pointer to object that is the key for the search
<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.

**19.45.2.15 qsort()** void qsort (  
     void \* base,  
     size\_t nmemb,  
     size\_t size,  
     int (\*)(const void \*, const void \*) \_\_reentrant compar )

Sort an array of **nmemb** items

#### Parameters

<i>base</i>	Pointer to first object in the array to sort
<i>nmemb</i>	Number of elements in the array
<i>size</i>	Size in bytes of each element in the array
<i>compar</i>	Function used to compare and sort two elements of the array

## 19.46 stdnoreturn.h File Reference

### Macros

- #define `noreturn` `_Noreturn`

### 19.46.1 Macro Definition Documentation

**19.46.1.1 noreturn** #define `noreturn` `_Noreturn`

## 19.47 string.h File Reference

#include <types.h>

### Functions

- char \* `strcpy` (char \*dest, const char \*src) `NONBANKED` \_\_preserves\_regs(b
- int `strcmp` (const char \*s1, const char \*s2) `NONBANKED` \_\_preserves\_regs(b
- void \* `memcpy` (void \*dest, const void \*src, size\_t len) `NONBANKED` \_\_preserves\_regs(b
- void \* `memmove` (void \*dest, const void \*src, size\_t n)
- void \* `memset` (void \*s, int c, size\_t n) `NONBANKED` \_\_preserves\_regs(b
- char \* `reverse` (char \*s) \_\_preserves\_regs(b
- char \* `strcat` (char \*s1, const char \*s2) `NONBANKED`
- int `strlen` (const char \*s) `NONBANKED` \_\_preserves\_regs(b
- 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`

### Variables

- char `c`

### 19.47.1 Detailed Description

Generic string functions.

## 19.47.2 Function Documentation

**19.47.2.1 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

**19.47.2.2 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**

**19.47.2.3 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

**19.47.2.4 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

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

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

#### Parameters

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

**19.47.2.6 reverse()** `char* reverse (`  
`char * s )`

Reverses the characters in a string

#### Parameters

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

For example 'abcdefg' will become 'gfedcba'.

Banked as the string must be modifiable.

Returns: Pointer to **s**

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

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

#### Parameters

<b>s1</b>	String to append onto
<b>s2</b>	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**

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

Calculates the length of a string

#### Parameters

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

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

**19.47.2.9 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**

**19.47.2.10 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**

**19.47.2.11 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**

### 19.47.3 Variable Documentation

**19.47.3.1 c** `int c`

## 19.48 time.h File Reference

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

### Macros

- `#define CLOCKS_PER_SEC 60`

### Typedefs

- `typedef UINT16 time_t`

### Functions

- `clock_t clock (void) NONBANKED`
- `time_t time (time_t *)`

#### 19.48.1 Detailed Description

Sort of ANSI compliant time functions.

#### 19.48.2 Macro Definition Documentation

**19.48.2.1 CLOCKS\_PER\_SEC** `#define CLOCKS_PER_SEC 60`

#### 19.48.3 Typedef Documentation

**19.48.3.1 time\_t** `typedef UINT16 time_t`

#### 19.48.4 Function Documentation

**19.48.4.1 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()`

**19.48.4.2 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\(\)](#)

## 19.49 `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`

### 19.49.1 Macro Definition Documentation

**19.49.1.1 `TYPEOF_INT`** `#define TYPEOF_INT 1`

**19.49.1.2 `TYPEOF_SHORT`** `#define TYPEOF_SHORT 2`

**19.49.1.3 `TYPEOF_CHAR`** `#define TYPEOF_CHAR 3`

**19.49.1.4 `TYPEOF_LONG`** `#define TYPEOF_LONG 4`

**19.49.1.5 `TYPEOF_FLOAT`** `#define TYPEOF_FLOAT 5`

**19.49.1.6 `TYPEOF_FIXED16X16`** `#define TYPEOF_FIXED16X16 6`

**19.49.1.7 `TYPEOF_BIT`** `#define TYPEOF_BIT 7`

**19.49.1.8    `typedef_BITFIELD`**    `#define typedef_BITFIELD 8`

**19.49.1.9    `typedef_SBIT`**    `#define typedef_SBIT 9`

**19.49.1.10   `typedef_SFR`**    `#define typedef_SFR 10`

**19.49.1.11   `typedef_VOID`**    `#define typedef_VOID 11`

**19.49.1.12   `typedef_STRUCT`**    `#define typedef_STRUCT 12`

**19.49.1.13   `typedef_ARRAY`**    `#define typedef_ARRAY 13`

**19.49.1.14   `typedef_FUNCTION`**    `#define typedef_FUNCTION 14`

**19.49.1.15   `typedef_POINTER`**    `#define typedef_POINTER 15`

**19.49.1.16   `typedef_FPOINTER`**    `#define typedef_FPOINTER 16`

**19.49.1.17   `typedef_CPOINTER`**    `#define typedef_CPOINTER 17`

**19.49.1.18   `typedef_GPOINTER`**    `#define typedef_GPOINTER 18`

**19.49.1.19   `typedef_PPOINTER`**    `#define typedef_PPOINTER 19`

**19.49.1.20   `typedef_IPOINTER`**    `#define typedef_IPOINTER 20`

**19.49.1.21   `typedef_EEPPPOINTER`**    `#define typedef_EEPPPOINTER 21`





## Index

/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/01\_getting\_started.md, 47  
ptr, 41  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/02\_regs\_and\_tools.md, 47  
segin, 42  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/03\_signifying\_gbdk.md, 47  
\_\_reentrant  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/04\_stdlib\_guidelines.md, 47  
\_\_render\_shadow\_OAM  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/05\_metasprites.md, 47  
\_\_setjmp  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/06\_tetrischain.md, 47  
\_\_cpu  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/07\_sample\_programs.md, 47  
\_\_current\_bank  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/08\_gb.h, 47  
\_\_fixed, 42  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/09\_migrating\_new\_versions.md, 47  
h, 42  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/10\_release\_notes.md, 47  
w, 42  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/20\_toolchain\_settings.md, 47  
gb.h, 105  
/home/birch/git/gbdev/gbdk2020/gbdk-2020-git/docs/pages/docs\_index.md, 47  
gb.h, 105  
\_\_GBDK\_VERSION  
gb.h, 77  
\_\_HandleCrash  
crash\_handler.h, 63  
\_\_PTRDIFF\_T\_DEFINED  
stddef.h, 126  
\_\_REG  
hardware.h, 109  
\_\_SIZE\_T\_DEFINED  
stddef.h, 126  
types.h, 49  
\_\_WCHAR\_T\_DEFINED  
stddef.h, 126  
\_\_assert  
assert.h, 52  
\_\_bool\_true\_false\_are\_defined  
stdbool.h, 125  
\_\_call\_banked  
far\_ptr.h, 71  
\_\_call\_banked\_addr  
far\_ptr.h, 71  
\_\_call\_banked\_bank  
far\_ptr.h, 71  
\_\_call\_banked\_ptr  
far\_ptr.h, 71  
\_\_current\_base\_tile  
metasprites.h, 117  
\_\_current\_metasprite  
metasprites.h, 117  
\_\_far\_ptr, 41  
fn, 42  
\_\_get\_started.md, 41  
ptr, 41  
\_\_regs\_and\_tools.md, 42  
segin, 42  
\_\_signifying\_gbdk.md, 47  
\_\_reentrant  
\_\_stdlib\_guidelines.md, 104  
\_\_render\_shadow\_OAM  
\_\_metasprites.md, 117  
\_\_setjmp  
\_\_tetrischain.md, 124  
\_\_cpu  
\_\_sample\_programs.md, 105  
\_\_current\_bank  
\_\_gb.h, 105  
\_\_fixed, 42  
\_\_migrating\_new\_versions.md, 42  
h, 42  
\_\_release\_notes.md, 42  
w, 42  
\_\_toolchain\_settings.md, 105  
gb.h, 105  
docs\_index.md, 105  
gb.h, 105  
\_\_io\_status  
gb.h, 105  
\_\_shadow\_OAM\_base  
gb.h, 106  
abs  
stdlib.h, 134  
add\_JOY  
gb.h, 85  
add\_LCD  
gb.h, 84  
add\_SIO  
gb.h, 85  
add\_TIM  
gb.h, 85  
add\_VBL  
gb.h, 84  
AND  
drawing.h, 65  
arand  
rand.h, 123  
asm/gbz80/provides.h, 47  
asm/gbz80/stdarg.h, 48  
asm/gbz80/types.h, 48  
asm/types.h, 50  
assert  
assert.h, 52  
assert.h, 51  
\_\_assert, 52  
assert, 52  
atoi

- stdlib.h, [135](#)
- atol
  - stdlib.h, [135](#)
- atomic\_flag, [42](#)
  - flag, [42](#)
- atomic\_flag\_clear
  - stdatomic.h, [125](#)
- atomic\_flag\_test\_and\_set
  - stdatomic.h, [125](#)
- b
  - \_fixed, [42](#)
  - gb.h, [106](#)
- BANKED
  - types.h, [49](#)
- BCD
  - bcd.h, [53](#)
- bcd.h, [52](#)
  - BCD, [53](#)
  - bcd2text, [53](#)
  - bcd\_add, [53](#)
  - BCD\_HEX, [52](#)
  - bcd\_sub, [53](#)
  - MAKE\_BCD, [52](#)
  - uint2bcd, [53](#)
- bcd2text
  - bcd.h, [53](#)
- bcd\_add
  - bcd.h, [53](#)
- BCD\_HEX
  - bcd.h, [52](#)
- bcd\_sub
  - bcd.h, [53](#)
- BCPD\_REG
  - hardware.h, [112](#)
- BCPS\_REG
  - hardware.h, [112](#)
- bgb\_emu.h
  - BGB\_MESSAGE, [56](#)
  - BGB\_MESSAGE\_FMT, [56](#)
  - BGB\_PROFILE\_BEGIN, [57](#)
  - BGB\_PROFILE\_END, [57](#)
  - BGB\_profiler\_message, [57](#)
  - BGB\_TEXT, [57](#)
- BGB\_MESSAGE
  - bgb\_emu.h, [56](#)
- BGB\_MESSAGE\_FMT
  - bgb\_emu.h, [56](#)
- BGB\_PROFILE\_BEGIN
  - bgb\_emu.h, [57](#)
- BGB\_PROFILE\_END
  - bgb\_emu.h, [57](#)
- BGB\_profiler\_message
  - bgb\_emu.h, [57](#)
- BGB\_TEXT
  - bgb\_emu.h, [57](#)
- BGP\_REG
  - hardware.h, [111](#)
- BLACK
  - drawing.h, [65](#)
- bool
  - stdbool.h, [125](#)
- BOOLEAN
  - types.h, [50](#)
- box
  - drawing.h, [67](#)
- BP\_SIZE
  - setjmp.h, [124](#)
- BPX\_SIZE
  - setjmp.h, [124](#)
- bsearch
  - stdlib.h, [136](#)
- BYTE
  - types.h, [50](#)
- c
  - gb.h, [105](#)
  - gbdecompress.h, [107](#)
  - string.h, [140](#)
- calloc
  - stdlib.h, [136](#)
- cgb.h
  - cgb\_compatibility, [62](#)
  - cpu\_fast, [62](#)
  - cpu\_slow, [61](#)
  - RGB, [58](#)
  - RGB\_AQUA, [59](#)
  - RGB\_BLACK, [59](#)
  - RGB\_BLUE, [59](#)
  - RGB\_BROWN, [60](#)
  - RGB\_CYAN, [59](#)
  - RGB\_DARKBLUE, [59](#)
  - RGB\_DARKGRAY, [59](#)
  - RGB\_DARKGREEN, [59](#)
  - RGB\_DARKRED, [59](#)
  - RGB\_DARKYELLOW, [59](#)
  - RGB\_GREEN, [59](#)
  - RGB\_LIGHTFLESH, [60](#)
  - RGB\_LIGHTGRAY, [59](#)
  - RGB\_ORANGE, [60](#)
  - RGB\_PINK, [59](#)
  - RGB\_PURPLE, [59](#)
  - RGB\_RED, [59](#)
  - RGB\_TEAL, [60](#)
  - RGB\_WHITE, [60](#)
  - RGB\_YELLOW, [59](#)
  - set\_bkg\_palette, [60](#)
  - set\_bkg\_palette\_entry, [61](#)
  - set\_sprite\_palette, [60](#)
  - set\_sprite\_palette\_entry, [61](#)
- cgb\_compatibility
  - cgb.h, [62](#)
- CGB\_TYPE
  - gb.h, [80](#)
- CHAR\_BIT
  - limits.h, [121](#)
- CHAR\_MAX
  - limits.h, [121](#)

CHAR\_MIN  
    limits.h, [121](#)  
circle  
    drawing.h, [67](#)  
clock  
    time.h, [141](#)  
clock\_t  
    types.h, [50](#)  
CLOCKS\_PER\_SEC  
    time.h, [141](#)  
cls  
    console.h, [63](#)  
color  
    drawing.h, [68](#)  
console.h  
    cls, [63](#)  
    gotoxy, [62](#)  
    posx, [62](#)  
    posy, [63](#)  
    setchar, [63](#)  
cpu\_fast  
    cgb.h, [62](#)  
cpu\_slow  
    cgb.h, [61](#)  
crash\_handler.h  
    \_\_HandleCrash, [63](#)  
CRITICAL  
    types.h, [49](#)  
ctype.h, [54](#)  
    isalpha, [54](#)  
    isdigit, [55](#)  
    islower, [54](#)  
    isspace, [55](#)  
    isupper, [54](#)  
    tolower, [55](#)  
    toupper, [55](#)  
  
d  
    gb.h, [106](#)  
debug  
    malloc.h, [113](#)  
delay  
    gb.h, [86](#)  
disable\_interrupts  
    gb.h, [88](#)  
DISABLE\_OAM\_DMA  
    gb.h, [83](#)  
DISABLE\_RAM\_MBC1  
    gb.h, [81](#)  
DISABLE\_RAM\_MBC5  
    gb.h, [82](#)  
DISPLAY\_OFF  
    gb.h, [82](#)  
display\_off  
    gb.h, [89](#)  
DISPLAY\_ON  
    gb.h, [82](#)  
DIV\_REG  
    hardware.h, [109](#)  
  
DKGREY  
    drawing.h, [65](#)  
DMA\_REG  
    hardware.h, [111](#)  
DMG\_TYPE  
    gb.h, [80](#)  
draw\_image  
    drawing.h, [67](#)  
drawing.h  
    AND, [65](#)  
    BLACK, [65](#)  
    box, [67](#)  
    circle, [67](#)  
    color, [68](#)  
    DKGREY, [65](#)  
    draw\_image, [67](#)  
    getpix, [67](#)  
    gotogxy, [68](#)  
    gprint, [65](#)  
    gprintf, [66](#)  
    gprintln, [65](#)  
    gprintn, [66](#)  
    GRAPHICS\_HEIGHT, [64](#)  
    GRAPHICS\_WIDTH, [64](#)  
    line, [67](#)  
    LTGREY, [65](#)  
    M\_FILL, [65](#)  
    M\_NOFILL, [65](#)  
    OR, [65](#)  
    plot, [66](#)  
    plot\_point, [67](#)  
    SIGNED, [65](#)  
    SOLID, [65](#)  
    switch\_data, [67](#)  
    UNSIGNED, [65](#)  
    WHITE, [65](#)  
    wrtchr, [67](#)  
    XOR, [65](#)  
  
dtile  
    metasprite\_t, [44](#)  
DWORD  
    types.h, [51](#)  
dx  
    metasprite\_t, [44](#)  
dy  
    metasprite\_t, [44](#)  
  
e  
    gb.h, [106](#)  
enable\_interrupts  
    gb.h, [88](#)  
ENABLE\_OAM\_DMA  
    gb.h, [83](#)  
ENABLE\_RAM\_MBC1  
    gb.h, [81](#)  
ENABLE\_RAM\_MBC5  
    gb.h, [82](#)  
exit  
    stdlib.h, [134](#)

FALSE  
     types.h, 51  
 false  
     stdbool.h, 125  
 FAR\_CALL  
     far\_ptr.h, 69  
 FAR\_FUNC  
     far\_ptr.h, 69  
 FAR\_OFS  
     far\_ptr.h, 69  
 FAR\_PTR  
     far\_ptr.h, 71  
 far\_ptr.h  
     \_\_call\_banked, 71  
     \_\_call\_banked\_addr, 71  
     \_\_call\_banked\_bank, 71  
     \_\_call\_banked\_ptr, 71  
     FAR\_CALL, 69  
     FAR\_FUNC, 69  
     FAR\_OFS, 69  
     FAR\_PTR, 71  
     FAR\_SEG, 69  
     TO\_FAR\_PTR, 69  
     to\_far\_ptr, 71  
 FAR\_SEG  
     far\_ptr.h, 69  
 fill\_bkg\_rect  
     gb.h, 104  
 fill\_win\_rect  
     gb.h, 105  
 first\_tile  
     sfont\_handle, 45  
 fixed  
     types.h, 51  
 flag  
     atomic\_flag, 42  
 fn  
     \_\_far\_ptr, 42  
 font  
     sfont\_handle, 45  
 font.h  
     FONT\_128ENCODING, 72  
     FONT\_256ENCODING, 72  
     FONT\_COMPRESSED, 72  
     font\_init, 73  
     font\_load, 73  
     FONT\_NOENCODING, 72  
     font\_set, 73  
     font\_t, 72  
     mfont\_handle, 73  
     pmfont\_handle, 73  
 FONT\_128ENCODING  
     font.h, 72  
 FONT\_256ENCODING  
     font.h, 72  
 FONT\_COMPRESSED  
     font.h, 72  
 font\_ibm  
     List of gbdk fonts, 41  
 font\_ibm\_fixed  
     List of gbdk fonts, 41  
 font\_init  
     font.h, 73  
 font\_italic  
     List of gbdk fonts, 41  
 font\_load  
     font.h, 73  
 font\_min  
     List of gbdk fonts, 41  
 FONT\_NOENCODING  
     font.h, 72  
 font\_set  
     font.h, 73  
 font\_spect  
     List of gbdk fonts, 41  
 font\_t  
     font.h, 72  
 free  
     stdlib.h, 136  
 gb.h  
     \_\_GBDK\_VERSION, 77  
     \_cpu, 105  
     \_current\_bank, 105  
     \_io\_in, 105  
     \_io\_out, 105  
     \_io\_status, 105  
     \_shadow\_OAM\_base, 106  
     add\_JOY, 85  
     add\_LCD, 84  
     add\_SIO, 85  
     add\_TIM, 85  
     add\_VBL, 84  
     b, 106  
     c, 105  
     CGB\_TYPE, 80  
     d, 106  
     delay, 86  
     disable\_interrupts, 88  
     DISABLE\_OAM\_DMA, 83  
     DISABLE\_RAM\_MBC1, 81  
     DISABLE\_RAM\_MBC5, 82  
     DISPLAY\_OFF, 82  
     display\_off, 89  
     DISPLAY\_ON, 82  
     DMG\_TYPE, 80  
     e, 106  
     enable\_interrupts, 88  
     ENABLE\_OAM\_DMA, 83  
     ENABLE\_RAM\_MBC1, 81  
     ENABLE\_RAM\_MBC5, 82  
     fill\_bkg\_rect, 104  
     fill\_win\_rect, 105  
     get\_bkg\_data, 90  
     get\_bkg\_tiles, 92  
     get\_bkg\_xy\_addr, 89  
     get\_data, 102

get\_mode, 86  
get\_sprite\_data, 99  
get\_sprite\_prop, 101  
get\_sprite\_tile, 100  
get\_tiles, 103  
get\_win\_data, 94  
get\_win\_tiles, 97  
get\_win\_xy\_addr, 94  
h, 105  
HIDE\_BKG, 82  
hide\_sprite, 102  
HIDE\_SPRITES, 83  
HIDE\_WIN, 83  
hramcpy, 89  
init\_bkg, 104  
init\_win, 104  
int\_handler, 83  
IO\_ERROR, 81  
IO\_IDLE, 80  
IO\_RECEIVING, 81  
IO\_SENDING, 81  
J\_A, 77  
J\_B, 77  
J\_DOWN, 77  
J\_LEFT, 77  
J\_RIGHT, 77  
J\_SELECT, 77  
J\_START, 77  
J\_UP, 77  
JOY\_IFLAG, 79  
joypad, 87  
joypad\_ex, 87  
joypad\_init, 87  
l, 106  
LCD\_IFLAG, 79  
M\_DRAWING, 78  
M\_NO\_INTERP, 78  
M\_NO\_SCROLL, 78  
M\_TEXT\_INOUT, 78  
M\_TEXT\_OUT, 78  
MAXWNDPOSX, 80  
MAXWNDPOSY, 80  
MGB\_TYPE, 80  
MINWNDPOSX, 80  
MINWNDPOSY, 80  
mode, 86  
move\_bkg, 93  
move\_sprite, 101  
move\_win, 98  
nowait\_int\_handler, 85  
OAM\_item\_t, 83  
receive\_byte, 86  
remove\_JOY, 84  
remove\_LCD, 84  
remove\_SIO, 84  
remove\_TIM, 84  
remove\_VBL, 83  
reset, 88  
S\_FLIPX, 78  
S\_FLIPY, 78  
S\_PALETTE, 78  
S\_PRIORITY, 78  
SCREENHEIGHT, 79  
SCREENWIDTH, 79  
scroll\_bkg, 93  
scroll\_sprite, 101  
scroll\_win, 98  
send\_byte, 86  
set\_bkg\_1bit\_data, 90  
set\_bkg\_data, 89  
set\_bkg\_submap, 92  
set\_bkg\_tile\_xy, 93  
set\_bkg\_tiles, 91  
set\_data, 102  
set\_interrupts, 88  
SET\_SHADOW\_OAM\_ADDRESS, 99  
set\_sprite\_1bit\_data, 99  
set\_sprite\_data, 98  
set\_sprite\_prop, 100  
set\_sprite\_tile, 100  
set\_tiles, 103  
set\_vram\_byte, 89  
set\_win\_1bit\_data, 94  
set\_win\_data, 94  
set\_win\_submap, 95  
set\_win\_tile\_xy, 97  
set\_win\_tiles, 95  
shadow\_OAM, 106  
SHOW\_BKG, 82  
SHOW\_SPRITES, 83  
SHOW\_WIN, 83  
SIO\_IFLAG, 79  
SPRITES\_8x16, 83  
SPRITES\_8x8, 83  
SWITCH\_16\_8\_MODE\_MBC1, 81  
SWITCH\_4\_32\_MODE\_MBC1, 81  
SWITCH\_RAM\_MBC1, 81  
SWITCH\_RAM\_MBC5, 82  
SWITCH\_ROM\_MBC1, 81  
SWITCH\_ROM\_MBC5, 81  
SWITCH\_ROM\_MBC5\_8M, 82  
sys\_time, 105  
TIM\_IFLAG, 79  
VBL\_IFLAG, 79  
vmemset, 104  
wait\_int\_handler, 85  
wait\_vbl\_done, 88  
waitpad, 87  
waitpadup, 87  
gb/bgb\_emu.h, 55  
gb/cgb.h, 57  
gb/console.h, 62  
gb/crash\_handler.h, 63  
gb/drawing.h, 63  
gb/far\_ptr.h, 68  
gb/font.h, 72

- gb/gb.h, [73](#)
- gb/gbdecompress.h, [106](#)
- gb/hardware.h, [108](#)
- gb/malloc.h, [112](#)
- gb/metasprites.h, [113](#)
- gb/sample.h, [117](#)
- gb/sgb.h, [118](#)
- gb\_decompress
  - gbdecompress.h, [106](#)
- gb\_decompress\_bkg\_data
  - gbdecompress.h, [106](#)
- gb\_decompress\_sprite\_data
  - gbdecompress.h, [107](#)
- gb\_decompress\_win\_data
  - gbdecompress.h, [107](#)
- gbdecompress.h
  - c, [107](#)
  - gb\_decompress, [106](#)
  - gb\_decompress\_bkg\_data, [106](#)
  - gb\_decompress\_sprite\_data, [107](#)
  - gb\_decompress\_win\_data, [107](#)
- gbdk-lib.h, [121](#)
- get\_bkg\_data
  - gb.h, [90](#)
- get\_bkg\_tiles
  - gb.h, [92](#)
- get\_bkg\_xy\_addr
  - gb.h, [89](#)
- get\_data
  - gb.h, [102](#)
- get\_mode
  - gb.h, [86](#)
- get\_sprite\_data
  - gb.h, [99](#)
- get\_sprite\_prop
  - gb.h, [101](#)
- get\_sprite\_tile
  - gb.h, [100](#)
- get\_tiles
  - gb.h, [103](#)
- get\_win\_data
  - gb.h, [94](#)
- get\_win\_tiles
  - gb.h, [97](#)
- get\_win\_xy\_addr
  - gb.h, [94](#)
- getchar
  - stdio.h, [133](#)
- getpix
  - drawing.h, [67](#)
- gets
  - stdio.h, [133](#)
- gotogxy
  - drawing.h, [68](#)
- gotoxy
  - console.h, [62](#)
- gprint
  - drawing.h, [65](#)
- gprintf
  - drawing.h, [66](#)
- gprintln
  - drawing.h, [65](#)
- gprintrn
  - drawing.h, [66](#)
- GRAPHICS\_HEIGHT
  - drawing.h, [64](#)
- GRAPHICS\_WIDTH
  - drawing.h, [64](#)
- h
  - \_fixed, [42](#)
  - gb.h, [105](#)
- hardware.h
  - \_\_REG, [109](#)
  - BCPD\_REG, [112](#)
  - BCPS\_REG, [112](#)
  - BGP\_REG, [111](#)
  - DIV\_REG, [109](#)
  - DMA\_REG, [111](#)
  - HDMA1\_REG, [111](#)
  - HDMA2\_REG, [111](#)
  - HDMA3\_REG, [111](#)
  - HDMA4\_REG, [111](#)
  - HDMA5\_REG, [112](#)
  - IE\_REG, [112](#)
  - IF\_REG, [109](#)
  - KEY1\_REG, [111](#)
  - LCDC\_REG, [111](#)
  - LY\_REG, [111](#)
  - LYC\_REG, [111](#)
  - NR10\_REG, [109](#)
  - NR11\_REG, [109](#)
  - NR12\_REG, [109](#)
  - NR13\_REG, [110](#)
  - NR14\_REG, [110](#)
  - NR21\_REG, [110](#)
  - NR22\_REG, [110](#)
  - NR23\_REG, [110](#)
  - NR24\_REG, [110](#)
  - NR30\_REG, [110](#)
  - NR31\_REG, [110](#)
  - NR32\_REG, [110](#)
  - NR33\_REG, [110](#)
  - NR34\_REG, [110](#)
  - NR41\_REG, [110](#)
  - NR42\_REG, [110](#)
  - NR43\_REG, [110](#)
  - NR44\_REG, [110](#)
  - NR50\_REG, [110](#)
  - NR51\_REG, [110](#)
  - NR52\_REG, [110](#)
  - OBP0\_REG, [111](#)
  - OBP1\_REG, [111](#)
  - OCPD\_REG, [112](#)
  - OCPS\_REG, [112](#)
  - P1\_REG, [109](#)
  - RP\_REG, [112](#)

SB\_REG, [109](#)  
SC\_REG, [109](#)  
SCX\_REG, [111](#)  
SCY\_REG, [111](#)  
STAT\_REG, [111](#)  
SVBK\_REG, [112](#)  
TAC\_REG, [109](#)  
TIMA\_REG, [109](#)  
TMA\_REG, [109](#)  
VBK\_REG, [111](#)  
WX\_REG, [111](#)  
WY\_REG, [111](#)  
HDMA1\_REG  
    [hardware.h](#), [111](#)  
HDMA2\_REG  
    [hardware.h](#), [111](#)  
HDMA3\_REG  
    [hardware.h](#), [111](#)  
HDMA4\_REG  
    [hardware.h](#), [111](#)  
HDMA5\_REG  
    [hardware.h](#), [112](#)  
HIDE\_BKG  
    [gb.h](#), [82](#)  
hide metasprite  
    [metasprites.h](#), [117](#)  
hide\_sprite  
    [gb.h](#), [102](#)  
HIDE\_SPRITES  
    [gb.h](#), [83](#)  
HIDE\_WIN  
    [gb.h](#), [83](#)  
hramcpy  
    [gb.h](#), [89](#)  
  
IE\_REG  
    [hardware.h](#), [112](#)  
IF\_REG  
    [hardware.h](#), [109](#)  
init\_bkg  
    [gb.h](#), [104](#)  
init\_win  
    [gb.h](#), [104](#)  
initarand  
    [rand.h](#), [123](#)  
initrand  
    [rand.h](#), [123](#)  
INT16  
    [types.h](#), [49](#)  
INT16\_C  
    [stdint.h](#), [130](#)  
INT16\_MAX  
    [stdint.h](#), [128](#)  
INT16\_MIN  
    [stdint.h](#), [128](#)  
int16\_t  
    [stdint.h](#), [131](#)  
INT32  
    [types.h](#), [49](#)  
  
INT32\_C  
    [stdint.h](#), [130](#)  
INT32\_MAX  
    [stdint.h](#), [128](#)  
INT32\_MIN  
    [stdint.h](#), [128](#)  
int32\_t  
    [stdint.h](#), [131](#)  
INT8  
    [types.h](#), [49](#)  
INT8\_C  
    [stdint.h](#), [130](#)  
INT8\_MAX  
    [stdint.h](#), [128](#)  
INT8\_MIN  
    [stdint.h](#), [128](#)  
int8\_t  
    [stdint.h](#), [131](#)  
INT\_FAST16\_MAX  
    [stdint.h](#), [129](#)  
INT\_FAST16\_MIN  
    [stdint.h](#), [129](#)  
int\_fast16\_t  
    [stdint.h](#), [131](#)  
INT\_FAST32\_MAX  
    [stdint.h](#), [129](#)  
INT\_FAST32\_MIN  
    [stdint.h](#), [129](#)  
int\_fast32\_t  
    [stdint.h](#), [131](#)  
INT\_FAST8\_MAX  
    [stdint.h](#), [129](#)  
INT\_FAST8\_MIN  
    [stdint.h](#), [129](#)  
int\_fast8\_t  
    [stdint.h](#), [131](#)  
int\_handler  
    [gb.h](#), [83](#)  
INT\_LEAST16\_MAX  
    [stdint.h](#), [128](#)  
INT\_LEAST16\_MIN  
    [stdint.h](#), [128](#)  
int\_least16\_t  
    [stdint.h](#), [131](#)  
INT\_LEAST32\_MAX  
    [stdint.h](#), [128](#)  
INT\_LEAST32\_MIN  
    [stdint.h](#), [128](#)  
int\_least32\_t  
    [stdint.h](#), [131](#)  
INT\_LEAST8\_MAX  
    [stdint.h](#), [128](#)  
INT\_LEAST8\_MIN  
    [stdint.h](#), [128](#)  
int\_least8\_t  
    [stdint.h](#), [131](#)  
INT\_MAX  
    [limits.h](#), [122](#)



INT\_MIN  
     limits.h, 121  
 INTERRUPT  
     types.h, 49  
 INTMAX\_C  
     stdint.h, 130  
 INTMAX\_MAX  
     stdint.h, 129  
 INTMAX\_MIN  
     stdint.h, 129  
 intmax\_t  
     stdint.h, 132  
 INTPTR\_MAX  
     stdint.h, 129  
 INTPTR\_MIN  
     stdint.h, 129  
 intptr\_t  
     stdint.h, 132  
 IO\_ERROR  
     gb.h, 81  
 IO\_IDLE  
     gb.h, 80  
 IO\_RECEIVING  
     gb.h, 81  
 IO\_SENDING  
     gb.h, 81  
 isalpha  
     ctype.h, 54  
 isdigit  
     ctype.h, 55  
 islower  
     ctype.h, 54  
 isspace  
     ctype.h, 55  
 isupper  
     ctype.h, 54  
 itoa  
     stdlib.h, 135  
  
 J\_A  
     gb.h, 77  
 J\_B  
     gb.h, 77  
 J\_DOWN  
     gb.h, 77  
 J\_LEFT  
     gb.h, 77  
 J\_RIGHT  
     gb.h, 77  
 J\_SELECT  
     gb.h, 77  
 J\_START  
     gb.h, 77  
 J\_UP  
     gb.h, 77  
 jmp\_buf  
     setjmp.h, 124  
 joy0  
     joypads\_t, 43  
 joy1  
     joypads\_t, 43  
 joy2  
     joypads\_t, 43  
 joy3  
     joypads\_t, 43  
 JOY\_IFLAG  
     gb.h, 79  
 joystick  
     gb.h, 87  
 joystick\_ex  
     gb.h, 87  
 joystick\_init  
     gb.h, 87  
 joypads  
     joypads\_t, 43  
 joypads\_t, 43  
     joy0, 43  
     joy1, 43  
     joy2, 43  
     joy3, 43  
     joypads, 43  
     npads, 43  
  
 KEY1\_REG  
     hardware.h, 111  
  
 l  
     \_fixed, 42  
     gb.h, 106  
 labs  
     stdlib.h, 134  
 LCD\_IFLAG  
     gb.h, 79  
 LCDC\_REG  
     hardware.h, 111  
 limits.h, 121  
     CHAR\_BIT, 121  
     CHAR\_MAX, 121  
     CHAR\_MIN, 121  
     INT\_MAX, 122  
     INT\_MIN, 121  
     LONG\_MAX, 122  
     LONG\_MIN, 122  
     SCHAR\_MAX, 121  
     SCHAR\_MIN, 121  
     SHRT\_MAX, 122  
     SHRT\_MIN, 122  
     UCHAR\_MAX, 121  
     UINT\_MAX, 122  
     UINT\_MIN, 122  
     ULONG\_MAX, 122  
     ULONG\_MIN, 122  
     USHRT\_MAX, 122  
     USHRT\_MIN, 122  
 line  
     drawing.h, 67  
 List of gbdk fonts, 40  
     font\_ibm, 41

- font\_ibm\_fixed, [41](#)
- font\_italic, [41](#)
- font\_min, [41](#)
- font\_spect, [41](#)
- LONG\_MAX
  - limits.h, [122](#)
- LONG\_MIN
  - limits.h, [122](#)
- longjmp
  - setjmp.h, [124](#)
- LTGREY
  - drawing.h, [65](#)
- ltoa
  - stdlib.h, [135](#)
- LWORD
  - types.h, [50](#)
- LY\_REG
  - hardware.h, [111](#)
- LYC\_REG
  - hardware.h, [111](#)
- M\_DRAWING
  - gb.h, [78](#)
- M\_FILL
  - drawing.h, [65](#)
- M\_NO\_INTERP
  - gb.h, [78](#)
- M\_NO\_SCROLL
  - gb.h, [78](#)
- M\_NOFILL
  - drawing.h, [65](#)
- M\_TEXT\_INOUT
  - gb.h, [78](#)
- M\_TEXT\_OUT
  - gb.h, [78](#)
- magic
  - smalloc\_hunk, [46](#)
- MAKE\_BCD
  - bcd.h, [52](#)
- malloc
  - stdlib.h, [136](#)
- malloc.h
  - debug, [113](#)
  - malloc\_first, [113](#)
  - MALLOC\_FREE, [113](#)
  - malloc\_gc, [113](#)
  - malloc\_heap\_start, [113](#)
  - MALLOC\_MAGIC, [113](#)
  - MALLOC\_USED, [113](#)
  - mmalloc\_hunk, [113](#)
  - pmmalloc\_hunk, [113](#)
- malloc\_first
  - malloc.h, [113](#)
- MALLOC\_FREE
  - malloc.h, [113](#)
- malloc\_gc
  - malloc.h, [113](#)
- malloc\_heap\_start
  - malloc.h, [113](#)
- MALLOC\_MAGIC
  - malloc.h, [113](#)
- MALLOC\_USED
  - malloc.h, [113](#)
- MAXWNDPOSX
  - gb.h, [80](#)
- MAXWNDPOSY
  - gb.h, [80](#)
- memcpy
  - string.h, [138](#)
- memmove
  - string.h, [138](#)
- memset
  - string.h, [139](#)
- metasprite\_end
  - metasprites.h, [114](#)
- metasprite\_t, [43](#)
  - dtile, [44](#)
  - dx, [44](#)
  - dy, [44](#)
  - metasprites.h, [114](#)
  - props, [44](#)
- metasprites.h
  - \_\_current\_base\_tile, [117](#)
  - \_\_current\_metasprite, [117](#)
  - \_\_render\_shadow\_OAM, [117](#)
  - hide\_metasprite, [117](#)
  - metasprite\_end, [114](#)
  - metasprite\_t, [114](#)
  - move\_metasprite, [115](#)
  - move\_metasprite\_hflip, [116](#)
  - move\_metasprite\_hvflip, [116](#)
  - move\_metasprite\_vflip, [115](#)
- mfont\_handle
  - font.h, [73](#)
- MGB\_TYPE
  - gb.h, [80](#)
- MINWNDPOSX
  - gb.h, [80](#)
- MINWNDPOSY
  - gb.h, [80](#)
- mmalloc\_hunk
  - malloc.h, [113](#)
- mode
  - gb.h, [86](#)
- move\_bkg
  - gb.h, [93](#)
- move\_metasprite
  - metasprites.h, [115](#)
- move\_metasprite\_hflip
  - metasprites.h, [116](#)
- move\_metasprite\_hvflip
  - metasprites.h, [116](#)
- move\_metasprite\_vflip
  - metasprites.h, [115](#)
- move\_sprite
  - gb.h, [101](#)
- move\_win

- gb.h, 98
- next
  - smalloc\_hunk, 46
- NONBANKED
  - types.h, 49
- noreturn
  - stdnoreturn.h, 137
- nowait\_int\_handler
  - gb.h, 85
- npads
  - joypads\_t, 43
- NR10\_REG
  - hardware.h, 109
- NR11\_REG
  - hardware.h, 109
- NR12\_REG
  - hardware.h, 109
- NR13\_REG
  - hardware.h, 110
- NR14\_REG
  - hardware.h, 110
- NR21\_REG
  - hardware.h, 110
- NR22\_REG
  - hardware.h, 110
- NR23\_REG
  - hardware.h, 110
- NR24\_REG
  - hardware.h, 110
- NR30\_REG
  - hardware.h, 110
- NR31\_REG
  - hardware.h, 110
- NR32\_REG
  - hardware.h, 110
- NR33\_REG
  - hardware.h, 110
- NR34\_REG
  - hardware.h, 110
- NR41\_REG
  - hardware.h, 110
- NR42\_REG
  - hardware.h, 110
- NR43\_REG
  - hardware.h, 110
- NR44\_REG
  - hardware.h, 110
- NR50\_REG
  - hardware.h, 110
- NR51\_REG
  - hardware.h, 110
- NR52\_REG
  - hardware.h, 110
- NULL
  - stddef.h, 126
  - types.h, 51
- OAM\_item\_t, 44
- gb.h, 83
- prop, 45
- tile, 45
- x, 45
- y, 45
- OBP0\_REG
  - hardware.h, 111
- OBP1\_REG
  - hardware.h, 111
- OCPD\_REG
  - hardware.h, 112
- OCPS\_REG
  - hardware.h, 112
- offsetof
  - stddef.h, 126
- ofs
  - \_\_far\_ptr, 41
- OR
  - drawing.h, 65
- P1\_REG
  - hardware.h, 109
- play\_sample
  - sample.h, 118
- plot
  - drawing.h, 66
- plot\_point
  - drawing.h, 67
- pmfont\_handle
  - font.h, 73
- pmmalloc\_hunk
  - malloc.h, 113
- POINTER
  - types.h, 51
- posix
  - console.h, 62
- posy
  - console.h, 63
- printf
  - stdio.h, 132
- prop
  - OAM\_item\_t, 45
- props
  - metasprite\_t, 44
- provides.h
  - USE\_C\_MEMCPY, 47
  - USE\_C\_STRCMP, 47
  - USE\_C\_STRCPY, 47
- ptr
  - \_\_far\_ptr, 41
- PTRDIFF\_MAX
  - stdint.h, 130
- PTRDIFF\_MIN
  - stdint.h, 129
- ptrdiff\_t
  - stddef.h, 126
- putchar
  - stdio.h, 132
- puts

- stdio.h, [133](#)
- qsort
  - stdlib.h, [137](#)
- rand
  - rand.h, [123](#)
- rand.h, [122](#)
  - arand, [123](#)
  - initarand, [123](#)
  - initrand, [123](#)
  - rand, [123](#)
  - randw, [123](#)
- randw
  - rand.h, [123](#)
- realloc
  - stdlib.h, [136](#)
- receive\_byte
  - gb.h, [86](#)
- remove\_JOY
  - gb.h, [84](#)
- remove\_LCD
  - gb.h, [84](#)
- remove\_SIO
  - gb.h, [84](#)
- remove\_TIM
  - gb.h, [84](#)
- remove\_VBL
  - gb.h, [83](#)
- reset
  - gb.h, [88](#)
- RET\_SIZE
  - setjmp.h, [124](#)
- reverse
  - string.h, [139](#)
- RGB
  - cgb.h, [58](#)
- RGB\_AQUA
  - cgb.h, [59](#)
- RGB\_BLACK
  - cgb.h, [59](#)
- RGB\_BLUE
  - cgb.h, [59](#)
- RGB\_BROWN
  - cgb.h, [60](#)
- RGB\_CYAN
  - cgb.h, [59](#)
- RGB\_DARKBLUE
  - cgb.h, [59](#)
- RGB\_DARKGRAY
  - cgb.h, [59](#)
- RGB\_DARKGREEN
  - cgb.h, [59](#)
- RGB\_DARKRED
  - cgb.h, [59](#)
- RGB\_DARKYELLOW
  - cgb.h, [59](#)
- RGB\_GREEN
  - cgb.h, [59](#)
- RGB\_LIGHTFLESH
  - cgb.h, [60](#)
- RGB\_LIGHTGRAY
  - cgb.h, [59](#)
- RGB\_ORANGE
  - cgb.h, [60](#)
- RGB\_PINK
  - cgb.h, [59](#)
- RGB\_PURPLE
  - cgb.h, [59](#)
- RGB\_RED
  - cgb.h, [59](#)
- RGB\_TEAL
  - cgb.h, [60](#)
- RGB\_WHITE
  - cgb.h, [60](#)
- RGB\_YELLOW
  - cgb.h, [59](#)
- RP\_REG
  - hardware.h, [112](#)
- S\_FLIPX
  - gb.h, [78](#)
- S\_FLIPY
  - gb.h, [78](#)
- S\_PALETTE
  - gb.h, [78](#)
- S\_PRIORITY
  - gb.h, [78](#)
- sample.h
  - play\_sample, [118](#)
- SB\_REG
  - hardware.h, [109](#)
- SC\_REG
  - hardware.h, [109](#)
- SCHAR\_MAX
  - limits.h, [121](#)
- SCHAR\_MIN
  - limits.h, [121](#)
- SCREENHEIGHT
  - gb.h, [79](#)
- SCREENWIDTH
  - gb.h, [79](#)
- scroll\_bkg
  - gb.h, [93](#)
- scroll\_sprite
  - gb.h, [101](#)
- scroll\_win
  - gb.h, [98](#)
- SCX\_REG
  - hardware.h, [111](#)
- SCY\_REG
  - hardware.h, [111](#)
- seg
  - \_\_far\_ptr, [41](#)
- segfn
  - \_\_far\_ptr, [42](#)
- segofs
  - \_\_far\_ptr, [41](#)

- send\_byte
  - gb.h, [86](#)
- set\_bkg\_1bit\_data
  - gb.h, [90](#)
- set\_bkg\_data
  - gb.h, [89](#)
- set\_bkg\_palette
  - cgb.h, [60](#)
- set\_bkg\_palette\_entry
  - cgb.h, [61](#)
- set\_bkg\_submap
  - gb.h, [92](#)
- set\_bkg\_tile\_xy
  - gb.h, [93](#)
- set\_bkg\_tiles
  - gb.h, [91](#)
- set\_data
  - gb.h, [102](#)
- set\_interrupts
  - gb.h, [88](#)
- SET\_SHADOW\_OAM\_ADDRESS
  - gb.h, [99](#)
- set\_sprite\_1bit\_data
  - gb.h, [99](#)
- set\_sprite\_data
  - gb.h, [98](#)
- set\_sprite\_palette
  - cgb.h, [60](#)
- set\_sprite\_palette\_entry
  - cgb.h, [61](#)
- set\_sprite\_prop
  - gb.h, [100](#)
- set\_sprite\_tile
  - gb.h, [100](#)
- set\_tiles
  - gb.h, [103](#)
- set\_vram\_byte
  - gb.h, [89](#)
- set\_win\_1bit\_data
  - gb.h, [94](#)
- set\_win\_data
  - gb.h, [94](#)
- set\_win\_submap
  - gb.h, [95](#)
- set\_win\_tile\_xy
  - gb.h, [97](#)
- set\_win\_tiles
  - gb.h, [95](#)
- setchar
  - console.h, [63](#)
- setjmp
  - setjmp.h, [124](#)
- setjmp.h, [123](#)
  - \_\_setjmp, [124](#)
  - BP\_SIZE, [124](#)
  - BPX\_SIZE, [124](#)
  - jmp\_buf, [124](#)
  - longjmp, [124](#)
  - RET\_SIZE, [124](#)
  - setjmp, [124](#)
  - SP\_SIZE, [124](#)
  - SPX\_SIZE, [124](#)
- sfont\_handle, [45](#)
  - first\_tile, [45](#)
  - font, [45](#)
- sgb.h
  - SGB\_ATTRC\_EN, [119](#)
  - SGB\_ATTR\_BLK, [119](#)
  - SGB\_ATTR\_CHR, [119](#)
  - SGB\_ATTR\_DIV, [119](#)
  - SGB\_ATTR\_LIN, [119](#)
  - SGB\_ATTR\_SET, [120](#)
  - SGB\_ATTR\_TRN, [120](#)
  - sgb\_check, [120](#)
  - SGB\_CHR\_TRN, [120](#)
  - SGB\_DATA\_SND, [119](#)
  - SGB\_DATA\_TRN, [119](#)
  - SGB\_ICON\_EN, [119](#)
  - SGB\_JUMP, [120](#)
  - SGB\_MASK\_EN, [120](#)
  - SGB\_MLT\_REQ, [119](#)
  - SGB\_OBJ\_TRN, [120](#)
  - SGB\_PAL\_01, [119](#)
  - SGB\_PAL\_03, [119](#)
  - SGB\_PAL\_12, [119](#)
  - SGB\_PAL\_23, [119](#)
  - SGB\_PAL\_SET, [119](#)
  - SGB\_PAL\_TRN, [119](#)
  - SGB\_PCT\_TRN, [120](#)
  - SGB\_SOU\_TRN, [119](#)
  - SGB\_SOUND, [119](#)
  - SGB\_TEST\_EN, [119](#)
  - sgb\_transfer, [120](#)
  - sgb\_transfer\_nowait, [120](#)
  - SGB\_ATTRC\_EN
    - sgb.h, [119](#)
  - SGB\_ATTR\_BLK
    - sgb.h, [119](#)
  - SGB\_ATTR\_CHR
    - sgb.h, [119](#)
  - SGB\_ATTR\_DIV
    - sgb.h, [119](#)
  - SGB\_ATTR\_LIN
    - sgb.h, [119](#)
  - SGB\_ATTR\_SET
    - sgb.h, [120](#)
  - SGB\_ATTR\_TRN
    - sgb.h, [120](#)
  - sgb\_check
    - sgb.h, [120](#)
  - SGB\_CHR\_TRN
    - sgb.h, [120](#)
  - SGB\_DATA\_SND
    - sgb.h, [119](#)
  - SGB\_DATA\_TRN
    - sgb.h, [119](#)

SGB\_ICON\_EN  
  sgb.h, 119

SGB\_JUMP  
  sgb.h, 120

SGB\_MASK\_EN  
  sgb.h, 120

SGB\_MLT\_REQ  
  sgb.h, 119

SGB\_OBJ\_TRN  
  sgb.h, 120

SGB\_PAL\_01  
  sgb.h, 119

SGB\_PAL\_03  
  sgb.h, 119

SGB\_PAL\_12  
  sgb.h, 119

SGB\_PAL\_23  
  sgb.h, 119

SGB\_PAL\_SET  
  sgb.h, 119

SGB\_PAL\_TRN  
  sgb.h, 119

SGB\_PCT\_TRN  
  sgb.h, 120

SGB\_SOU\_TRN  
  sgb.h, 119

SGB\_SOUND  
  sgb.h, 119

SGB\_TEST\_EN  
  sgb.h, 119

sgb\_transfer  
  sgb.h, 120

sgb\_transfer\_nowait  
  sgb.h, 120

shadow\_OAM  
  gb.h, 106

SHOW\_BKG  
  gb.h, 82

SHOW\_SPRITES  
  gb.h, 83

SHOW\_WIN  
  gb.h, 83

SHRT\_MAX  
  limits.h, 122

SHRT\_MIN  
  limits.h, 122

SIG\_ATOMIC\_MAX  
  stdint.h, 130

SIG\_ATOMIC\_MIN  
  stdint.h, 130

SIGNED  
  drawing.h, 65

SIO\_IFLAG  
  gb.h, 79

size  
  smalloc\_hunk, 46

SIZE\_MAX  
  stdint.h, 130

size\_t  
  stddef.h, 126  
  types.h, 49

smalloc\_hunk, 45  
  magic, 46  
  next, 46  
  size, 46  
  status, 46

SOLID  
  drawing.h, 65

SP\_SIZE  
  setjmp.h, 124

sprintf  
  stdio.h, 133

SPRITES\_8x16  
  gb.h, 83

SPRITES\_8x8  
  gb.h, 83

SPX\_SIZE  
  setjmp.h, 124

STAT\_REG  
  hardware.h, 111

status  
  smalloc\_hunk, 46

stdarg.h, 48  
  va\_arg, 48  
  va\_end, 48  
  va\_list, 48  
  va\_start, 48

stdatomic.h, 125  
  atomic\_flag\_clear, 125  
  atomic\_flag\_test\_and\_set, 125

stdbool.h, 125  
  \_\_bool\_true\_false\_are\_defined, 125  
  bool, 125  
  false, 125  
  true, 125

stddef.h, 125  
  \_\_PTRDIFF\_T\_DEFINED, 126  
  \_\_SIZE\_T\_DEFINED, 126  
  \_\_WCHAR\_T\_DEFINED, 126  
  NULL, 126  
  offsetof, 126  
  ptrdiff\_t, 126  
  size\_t, 126  
  wchar\_t, 126

stdint.h, 126  
  INT16\_C, 130  
  INT16\_MAX, 128  
  INT16\_MIN, 128  
  int16\_t, 131  
  INT32\_C, 130  
  INT32\_MAX, 128  
  INT32\_MIN, 128  
  int32\_t, 131  
  INT8\_C, 130  
  INT8\_MAX, 128  
  INT8\_MIN, 128

int8\_t, 131  
 INT\_FAST16\_MAX, 129  
 INT\_FAST16\_MIN, 129  
 int\_fast16\_t, 131  
 INT\_FAST32\_MAX, 129  
 INT\_FAST32\_MIN, 129  
 int\_fast32\_t, 131  
 INT\_FAST8\_MAX, 129  
 INT\_FAST8\_MIN, 129  
 int\_fast8\_t, 131  
 INT\_LEAST16\_MAX, 128  
 INT\_LEAST16\_MIN, 128  
 int\_least16\_t, 131  
 INT\_LEAST32\_MAX, 128  
 INT\_LEAST32\_MIN, 128  
 int\_least32\_t, 131  
 INT\_LEAST8\_MAX, 128  
 INT\_LEAST8\_MIN, 128  
 int\_least8\_t, 131  
 INTMAX\_C, 130  
 INTMAX\_MAX, 129  
 INTMAX\_MIN, 129  
 intmax\_t, 132  
 INTPTR\_MAX, 129  
 INTPTR\_MIN, 129  
 intptr\_t, 132  
 PTRDIFF\_MAX, 130  
 PTRDIFF\_MIN, 129  
 SIG\_ATOMIC\_MAX, 130  
 SIG\_ATOMIC\_MIN, 130  
 SIZE\_MAX, 130  
 UINT16\_C, 130  
 UINT16\_MAX, 128  
 uint16\_t, 131  
 UINT32\_C, 130  
 UINT32\_MAX, 128  
 uint32\_t, 131  
 UINT8\_C, 130  
 UINT8\_MAX, 128  
 uint8\_t, 131  
 UINT\_FAST16\_MAX, 129  
 uint\_fast16\_t, 131  
 UINT\_FAST32\_MAX, 129  
 uint\_fast32\_t, 132  
 UINT\_FAST8\_MAX, 129  
 uint\_fast8\_t, 131  
 UINT\_LEAST16\_MAX, 129  
 uint\_least16\_t, 131  
 UINT\_LEAST32\_MAX, 129  
 uint\_least32\_t, 131  
 UINT\_LEAST8\_MAX, 128  
 uint\_least8\_t, 131  
 UINTMAX\_C, 130  
 UINTMAX\_MAX, 129  
 uintmax\_t, 132  
 UINTPTR\_MAX, 129  
 uintptr\_t, 132  
 WCHAR\_MAX, 130  
 WCHAR\_MIN, 130  
 WINT\_MAX, 130  
 WINT\_MIN, 130  
 stdio.h, 132  
   getchar, 133  
   gets, 133  
   printf, 132  
   putchar, 132  
   puts, 133  
   sprintf, 133  
 stdlib.h, 133  
   \_\_reentrant, 134  
   abs, 134  
   atoi, 135  
   atol, 135  
   bsearch, 136  
   calloc, 136  
   exit, 134  
   free, 136  
   itoa, 135  
   labs, 134  
   ltoa, 135  
   malloc, 136  
   qsort, 137  
   realloc, 136  
   ultoa, 136  
   utoa, 135  
 stdnoreturn.h, 137  
   noreturn, 137  
 strcat  
   string.h, 139  
 strcmp  
   string.h, 138  
 strcpy  
   string.h, 138  
 string.h, 137  
   c, 140  
   memcpy, 138  
   memmove, 138  
   memset, 139  
   reverse, 139  
   strcat, 139  
   strcmp, 138  
   strcpy, 138  
   strlen, 139  
   strncat, 139  
   strncmp, 140  
   strncpy, 140  
 strlen  
   string.h, 139  
 strncat  
   string.h, 139  
 strncmp  
   string.h, 140  
 strncpy  
   string.h, 140  
 SVBK\_REG  
   hardware.h, 112

SWITCH\_16\_8\_MODE\_MBC1  
     gb.h, 81  
 SWITCH\_4\_32\_MODE\_MBC1  
     gb.h, 81  
 switch\_data  
     drawing.h, 67  
 SWITCH\_RAM\_MBC1  
     gb.h, 81  
 SWITCH\_RAM\_MBC5  
     gb.h, 82  
 SWITCH\_ROM\_MBC1  
     gb.h, 81  
 SWITCH\_ROM\_MBC5  
     gb.h, 81  
 SWITCH\_ROM\_MBC5\_8M  
     gb.h, 82  
 sys\_time  
     gb.h, 105  
  
 TAC\_REG  
     hardware.h, 109  
 tile  
     OAM\_item\_t, 45  
 TIM\_IFLAG  
     gb.h, 79  
 TIMA\_REG  
     hardware.h, 109  
 time  
     time.h, 141  
 time.h, 141  
     clock, 141  
     CLOCKS\_PER\_SEC, 141  
     time, 141  
     time\_t, 141  
 time\_t  
     time.h, 141  
 TMA\_REG  
     hardware.h, 109  
 TO\_FAR\_PTR  
     far\_ptr.h, 69  
 to\_far\_ptr  
     far\_ptr.h, 71  
 tolower  
     ctype.h, 55  
 toupper  
     ctype.h, 55  
 TRUE  
     types.h, 51  
 true  
     stdbool.h, 125  
 typeof.h, 142  
     TYPEOF\_ARRAY, 143  
     TYPEOF\_BIT, 142  
     TYPEOF\_BITFIELD, 142  
     TYPEOF\_CHAR, 142  
     TYPEOF\_CPOINTER, 143  
     TYPEOF\_EEPPPOINTER, 143  
     TYPEOF\_FIXED16X16, 142  
     TYPEOF\_FLOAT, 142  
     TYPEOF\_FPOINTER, 143  
     TYPEOF\_FUNCTION, 143  
     TYPEOF\_GPOINTER, 143  
     TYPEOF\_INT, 142  
     TYPEOF\_IPOINTER, 143  
     TYPEOF\_LONG, 142  
     TYPEOF\_POINTER, 143  
     TYPEOF\_PPOINTER, 143  
     TYPEOF\_SBIT, 143  
     TYPEOF\_SFR, 143  
     TYPEOF\_SHORT, 142  
     TYPEOF\_STRUCT, 143  
     TYPEOF\_VOID, 143  
 TYPEOF\_ARRAY  
     typeof.h, 143  
 TYPEOF\_BIT  
     typeof.h, 142  
 TYPEOF\_BITFIELD  
     typeof.h, 142  
 TYPEOF\_CHAR  
     typeof.h, 142  
 TYPEOF\_CPOINTER  
     typeof.h, 143  
 TYPEOF\_EEPPPOINTER  
     typeof.h, 143  
 TYPEOF\_FIXED16X16  
     typeof.h, 142  
 TYPEOF\_FLOAT  
     typeof.h, 142  
 TYPEOF\_FPOINTER  
     typeof.h, 143  
 TYPEOF\_FUNCTION  
     typeof.h, 143  
 TYPEOF\_GPOINTER  
     typeof.h, 143  
 TYPEOF\_INT  
     typeof.h, 142  
 TYPEOF\_IPOINTER  
     typeof.h, 143  
 TYPEOF\_LONG  
     typeof.h, 142  
 TYPEOF\_POINTER  
     typeof.h, 143  
 TYPEOF\_PPOINTER  
     typeof.h, 143  
 TYPEOF\_SBIT  
     typeof.h, 143  
 TYPEOF\_SFR  
     typeof.h, 143  
 TYPEOF\_SHORT  
     typeof.h, 142  
 TYPEOF\_STRUCT  
     typeof.h, 143  
 TYPEOF\_VOID  
     typeof.h, 143  
 types.h, 51  
     \_\_SIZE\_T\_DEFINED, 49  
     BANKED, 49



- BOOLEAN, [50](#)
- BYTE, [50](#)
- clock\_t, [50](#)
- CRITICAL, [49](#)
- DWORD, [51](#)
- FALSE, [51](#)
- fixed, [51](#)
- INT16, [49](#)
- INT32, [49](#)
- INT8, [49](#)
- INTERRUPT, [49](#)
- LWORD, [50](#)
- NONBANKED, [49](#)
- NULL, [51](#)
- POINTER, [51](#)
- size\_t, [49](#)
- TRUE, [51](#)
- UBYTE, [50](#)
- UDWORD, [51](#)
- UINT16, [49](#)
- UINT32, [49](#)
- UINT8, [49](#)
- ULWORD, [50](#)
- UWORD, [50](#)
- WORD, [50](#)
- UBYTE
  - types.h, [50](#)
- UCHAR\_MAX
  - limits.h, [121](#)
- UDWORD
  - types.h, [51](#)
- UINT16
  - types.h, [49](#)
- UINT16\_C
  - stdint.h, [130](#)
- UINT16\_MAX
  - stdint.h, [128](#)
- uint16\_t
  - stdint.h, [131](#)
- uint2bcd
  - bcd.h, [53](#)
- UINT32
  - types.h, [49](#)
- UINT32\_C
  - stdint.h, [130](#)
- UINT32\_MAX
  - stdint.h, [128](#)
- uint32\_t
  - stdint.h, [131](#)
- UINT8
  - types.h, [49](#)
- UINT8\_C
  - stdint.h, [130](#)
- UINT8\_MAX
  - stdint.h, [128](#)
- uint8\_t
  - stdint.h, [131](#)
- UINT\_FAST16\_MAX
  - stdint.h, [129](#)
- uint\_fast16\_t
  - stdint.h, [131](#)
- UINT\_FAST32\_MAX
  - stdint.h, [129](#)
- uint\_fast32\_t
  - stdint.h, [132](#)
- UINT\_FAST8\_MAX
  - stdint.h, [129](#)
- uint\_fast8\_t
  - stdint.h, [131](#)
- UINT\_LEAST16\_MAX
  - stdint.h, [129](#)
- uint\_least16\_t
  - stdint.h, [131](#)
- UINT\_LEAST32\_MAX
  - stdint.h, [129](#)
- uint\_least32\_t
  - stdint.h, [131](#)
- UINT\_LEAST8\_MAX
  - stdint.h, [128](#)
- uint\_least8\_t
  - stdint.h, [131](#)
- UINT\_MAX
  - limits.h, [122](#)
- UINT\_MIN
  - limits.h, [122](#)
- UINTMAX\_C
  - stdint.h, [130](#)
- UINTMAX\_MAX
  - stdint.h, [129](#)
- uintmax\_t
  - stdint.h, [132](#)
- UINTPTR\_MAX
  - stdint.h, [129](#)
- uintptr\_t
  - stdint.h, [132](#)
- ULONG\_MAX
  - limits.h, [122](#)
- ULONG\_MIN
  - limits.h, [122](#)
- ultoa
  - stdlib.h, [136](#)
- ULWORD
  - types.h, [50](#)
- UNSIGNED
  - drawing.h, [65](#)
- USE\_C\_MEMCPY
  - provides.h, [47](#)
- USE\_C\_STRCMP
  - provides.h, [47](#)
- USE\_C\_STRCPY
  - provides.h, [47](#)
- USHRT\_MAX
  - limits.h, [122](#)
- USHRT\_MIN
  - limits.h, [122](#)
- utoa

- stdlib.h, [135](#)
- UWORD
  - types.h, [50](#)
- va\_arg
  - stdarg.h, [48](#)
- va\_end
  - stdarg.h, [48](#)
- va\_list
  - stdarg.h, [48](#)
- va\_start
  - stdarg.h, [48](#)
- VBK\_REG
  - hardware.h, [111](#)
- VBL\_IFLAG
  - gb.h, [79](#)
- vmemset
  - gb.h, [104](#)
- w
  - \_fixed, [42](#)
- wait\_int\_handler
  - gb.h, [85](#)
- wait\_vbl\_done
  - gb.h, [88](#)
- waitpad
  - gb.h, [87](#)
- waitpadup
  - gb.h, [87](#)
- WCHAR\_MAX
  - stdint.h, [130](#)
- WCHAR\_MIN
  - stdint.h, [130](#)
- wchar\_t
  - stddef.h, [126](#)
- WHITE
  - drawing.h, [65](#)
- WINT\_MAX
  - stdint.h, [130](#)
- WINT\_MIN
  - stdint.h, [130](#)
- WORD
  - types.h, [50](#)
- wrtchr
  - drawing.h, [67](#)
- WX\_REG
  - hardware.h, [111](#)
- WY\_REG
  - hardware.h, [111](#)
- x
  - OAM\_item\_t, [45](#)
- XOR
  - drawing.h, [65](#)
- y
  - OAM\_item\_t, [45](#)