NAME

rgblink — Game Boy linker

SYNOPSIS

rgblink [-dhMtVvwx] [-l *linker_script*] [-m *map_file*] [-n *sym_file*]
[-O *overlay_file*] [-o *out_file*] [-p *pad_value*] [-S *spec*] [-W *warning*]
file ...

DESCRIPTION

The **rgblink** program links RGB object files, typically created by *rgbasm*(1), into a single Game Boy ROM file. The object file format is documented in *rgbds*(5).

ROM0 sections are placed in the first 16 KiB of the output ROM, and ROMX sections are placed in any 16 KiB "bank" except the first. If your ROM will only be 32 KiB, you can use the `-t` option to change this.

Similarly, WRAM0 sections are placed in the first 4 KiB of WRAM ("bank 0"), and WRAMX sections are placed in any bank of the last 4 KiB. If your ROM doesn't use banked WRAM, you can use the `-w` option to change this.

Also, if your ROM is designed for a monochrome Game Boy, you can make sure that you don't use any incompatible section by using the `-d` option, which implies `-w` but also prohibits the use of banked VRAM.

The input *asmfile* can be a path to a file, or `-` to read from standard input.

Note that options can be abbreviated as long as the abbreviation is unambiguous: `--verb` is `--verbose`, but `--ver` is invalid because it could also be `--version`. The arguments are as follows:

`-d`, `--dmg`
: Enable DMG mode. Prohibit the use of sections that doesn't exist on a DMG, such as VRAM bank 1. This option automatically enables `-w`.

`-h`, `--help`
: Print help text for the program and exit.

`-l` *linker_script*, `--linkerscript` *linker_script*
: Specify a linker script file that tells the linker how sections must be placed in the ROM. The attributes assigned in the linker script must be consistent with any assigned in the code. See *rgblink*(5) for more information about the linker script format.

`-M`, `--no-sym-in-map`
: If specified, the map file will not list symbols, only sections.

`-m` *map_file*, `--map` *map_file*
: Write a map file to the given filename, listing how sections and symbols were assigned.

`-n` *sym_file*, `--sym` *sym_file*
: Write a symbol file to the given filename, listing all visible labels and exported numeric constants. Labels output their bank and address, numeric constants output their value, following *this specification*: https://rgbds.gbdev.io/sym/. Several external programs can use this information, for example to help debugging ROMs.

`-O` *overlay_file*, `--overlay` *overlay_file*
: If specified, sections will be overlaid "on top" of the ROM image *overlay_file*: empty space between sections will be filled by the corresponding bytes from *overlay_file*. This is useful to patch an existing ROM. Note that all sections must be fixed (forced bank **and** address)!

`-o` *out_file*, `--output` *out_file*
: Write the ROM image to the given file.

`-p` *pad_value*, `--pad` *pad_value*
: When inserting padding between sections, pad with this value. The default is 0.

`-S` *spec*, `--scramble` *spec*
: Enables a different “scrambling” algorithm for placing sections. See “Scrambling algorithm” below for an explanation and a description of *spec*.

`-t`, `--tiny`
: Expand the ROM0 section size from 16 KiB to the full 32 KiB assigned to ROM. ROMX sections that are fixed to a bank other than 1 become errors, other ROMX sections are treated as ROM0. Useful for ROMs that fit in 32 KiB.

`-V`, `--version`
: Print the version of the program and exit.

`-v`, `--verbose`
: Verbose: enable printing more information to standard error.

`-W` *warning*, `--warning` *warning*
: Set warning flag *warning*. A warning message will be printed if *warning* is an unknown warning flag. See the “DIAGNOSTICS” section for a list of warnings.

`-w`, `--wramx`
: Expand the WRAM0 section size from 4 KiB to the full 8 KiB assigned to WRAM. WRAMX sections that are fixed to a bank other than 1 become errors, other WRAMX sections are treated as WRAM0.

`-x`, `--nopad`
: Disables padding the end of the final file. This option automatically enables `-t`. You can use this to make binary files that are not a ROM. When making a ROM, note that not using this is not a replacement for *rgbfix*(1)’s `-p` option!

Scrambling algorithm

The default section placement algorithm tries to minimize the number of banks used; “scrambling” instead places sections into a given pool of banks, trying to minimize the number of sections sharing a given bank. This is useful to catch broken bank assumptions, such as expecting two different sections to land in the same bank (that is not guaranteed unless both are manually assigned the same bank number).

A scrambling spec is a comma-separated list of region specs. A trailing comma is allowed, as well as whitespace between all specs and their components. Each region spec has the following form:

region[=*size*]

region must be one of the following (case-insensitive), while *size* must be a positive decimal integer between 1 and the corresponding maximum. Certain regions allow omitting the size, in which case it defaults to its max value.

Region name	Max size	Size optional
`romx`	65535	No
`sram`	255	No
`wramx`	7	Yes

A *size* of 0 disables scrambling for that region.

For example, `romx=64,wramx=4` will scramble **ROMX** sections among ROM banks 1 to 64, **WRAMX** sections among RAM banks 1 to 4, and will not scramble **SRAM** sections.

Later region specs override earlier ones; for example, `romx=42, Romx=0` disables scrambling for `romx`.

`wramx` scrambling is silently ignored if `-w` is passed (including if implied by `-d`), as **WRAMX** sections will be treated as **WRAM0**.

DIAGNOSTICS

Warnings are diagnostic messages that indicate possibly erroneous behavior that does not necessarily compromise the linking process. The following options alter the way warnings are processed.

`-Werror`
: Make all warnings into errors. This can be negated as `-Wno-error` to prevent turning all warnings into errors.

`-Werror=`
: Make the specified warning or meta warning into an error. A warning's name is appended (example: `-Werror=assert`), and this warning is implicitly enabled and turned into an error. This can be negated as `-Wno-error=` to prevent turning a specified warning into an error, even if `-Werror` is in effect.

The following warnings are "meta" warnings, that enable a collection of other warnings. If a specific warning is toggled via a meta flag and a specific one, the more specific one takes priority. The position on the command-line acts as a tie breaker, the last one taking effect.

`-Wall`
: This enables warnings that are likely to indicate an error or undesired behavior, and that can easily be fixed.

`-Weverything`
: Enables literally every warning.

The following warnings are actual warning flags; with each description, the corresponding warning flag is included. Note that each of these flag also has a negation (for example, `-Wobsolete` enables the warning that `-Wno-obsolete` disables; and `-Wall` enables every warning that `-Wno-all` disables). Only the non-default flag is listed here. Ignoring the "no-" prefix, entries are listed alphabetically.

`-Wno-assert`
: Warn when **WARN**-type assertions fail. (See "Aborting the assembly process" in *rgbasm*(5) for **ASSERT**).

`-Wdiv`
: Warn when dividing the smallest negative integer (-2**31) by -1, which yields itself due to integer overflow. This warning is enabled by `-Wall`.

`-Wno-obsolete`
: Warn when obsolete features are encountered, which have been deprecated and may later be removed.

`-Wshift`
: Warn when shifting right a negative value. Use a division by 2**N instead. This warning is enabled by `-Wall`.

`-Wshift-amount`
: Warn when a shift's operand is negative or greater than 32. This warning is enabled by `-Wall`.

`-Wno-truncation`
: Warn when an implicit truncation (for example, **db** to an 8-bit value) loses some bits. This occurs when an N-bit value is 2**N or greater, or less than -2**N.

EXAMPLES

All you need for a basic ROM is an object file, which can be made into a ROM image like so:

```
$ rgblink -o bar.gb foo.o
```

The resulting *bar.gb* will not have correct checksums (unless you put them in the assembly source). You should use *rgbfix*(1) to fix these so that the program will actually run in a Game Boy:

```
$ rgbfix -v bar.gb
```

Here is a more complete example:

```
$ rgblink -o bin/game.gb -n bin/game.sym -p 0xFF obj/title.o
obj/engine.o
```

BUGS

Please report bugs on *GitHub*: https://github.com/gbdev/rgbds/issues.

SEE ALSO

rgbasm(1), *rgblink*(5), *rgbfix*(1), *rgbgfx*(1), *gbz80*(7), *rgbds*(5), *rgbds*(7)

HISTORY

rgblink was originally written by Carsten Sørensen as part of the ASMotor package, and was later repackaged in RGBDS by Justin Lloyd. It is now maintained by a number of contributors at https://github.com/gbdev/rgbds.