NAME

rgblink — linker script file format

DESCRIPTION

The linker script is a file that allows specifying attributes for sections at link time, and in a centralized manner. There can only be one linker script per invocation of **rgblink**, but it can be split into several files (using the **INCLUDE** directive).

Basic syntax

The linker script syntax is line-based. Each line may have a directive or section name, a comment, both, or neither. Whitespace (space and tab characters) is used to separate syntax elements, but is otherwise ignored.

Comments begin with a semicolon ';' character, until the end of the line. They are simply ignored.

Keywords are composed of letters and digits (but they can't start with a digit); they are all case-insensitive.

Numbers can be written in a number of formats.

Format type	Possible prefixes	Accepted characters
Decimal	none	0123456789
Hexadecimal	`$`, `0x`, `0X`	0123456789ABCDEF
Octal	`&`, `0o`, `0O`	01234567
Binary	`%`, `0b`, `0B`	01

Underscores are also accepted in numbers, except at the beginning of one. This can be useful for grouping digits, like `1_234` or `$ff_80`.

Strings begin with a double quote, and end at the next (non-escaped) double quote. Strings must not contain literal newline characters. Most of the same character escapes as *rgbasm*(5) are supported, specifically '`\\`', '`\"`', '`\n`', '`\r`', '`\t`', and '`\0`'. Other backslash escape sequences in*r gbasm*(5) are only relevant to assembly code and do not apply in linker scripts.

Directives

Including other files

INCLUDE *path* acts as if the contents of the file at *path* were copy-pasted in place of the **INCLUDE** directive. *path* must be a string.

Specifying the active bank

The active bank can be set by specifying its type (memory region) and number. The possible types are: **ROM0**, **ROMX**, **VRAM**, **SRAM**, **WRAM0**, **WRAMX**, **OAM**, and **HRAM**. The bank number can be omitted from the types that only contain a single bank, which are: **ROM0**, **ROMX** if `-t` is passed to *rgblink*(1), **VRAM** if `-d` is passed to *rgblink*(1), **WRAM0**, **WRAMX** if `-w` is passed to *rgblink*(1), **OAM**, and **HRAM**. (**SRAM** is the only type that can never have its bank number omitted.)

After a bank specification, the "current address" is set to the last value it had for that bank. If the bank has never been active thus far, the "current address" defaults to the beginning of the bank (e.g. *$4000* for **ROMX** sections).

Instead of giving a bank number, the keyword **FLOATING** can be used instead; this sets the type of the subsequent sections without binding them to a particular bank. (If the type only allows a single bank, e.g. **ROM0**, then **FLOATING** is v alid but redundant and has no effect.) Since no particular section is active, the "current address" is made floating (as if by a **FLOATING** directive), and **ORG** is not allowed.

Changing the current address

A bank must be active for any of these directives to be used.

ORG *addr* sets the "current address" to *addr*. This directive cannot be used to move the address backwards: *addr* must be greater than or equal to the "current address".

FLOATING causes all sections between it and the next **ORG** or bank specification to be placed at addresses automatically determined by **rgblink**. (It is, however, compatible with **ALIGN** below.)

ALIGN *addr*, *offset* increases the "current address" until it is aligned to the specified boundary (i.e. the *align* lowest bits of the address are equal to *offset*). If*offset* is omitted, it is implied to be 0. For example, if the "current address" is $0007, `ALIGN 8` would set it to $0100, and `ALIGN 8, 10` would set it to $000A.

DS *size* increases the "current address" by *size*. The gap is not allocated, so smaller floating sections can later be placed there.

Section placement

A section can be placed simply by naming it (with a string). Its bank is set to the active bank, and its address to the "current address". Any constraints the section already possesses (whether from earlier in the linker script, or from the object files being linked) must be consistent with what the linker script specifies: the section's type must match, the section's bank number (if set) must match the active bank, etc. In particular, if the section has an alignment constraint, the address at which it is placed by the linker script must obey that constraint; otherwise, an error will occur.

After a section is placed, the "current address" is increased by the section's size. This must not increase it past the end of the active memory region.

The section must have been defined in the object files being linked, unless the section name is followed by the keyword **OPTIONAL**.

EXAMPLES

```
; This line contains only a comment
ROMX $F            ; start a bank
  "Some functions" ; a section name
  ALIGN 8          ; a directive
  "Some \"array\""

WRAMX 2            ; start another bank
  org $d123        ; another directive
  "Some variables"
```

SEE ALSO

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

HISTORY

rgblink(1) 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.