NAME

rgbds — object file format documentation

DESCRIPTION

This is the description of the object files used by rgbasm(1) and rgblink(1). *Please note that the specifications may change.* This toolchain is in development and new features may require adding more information to the current format, or modifying some fields, which would break compatibility with older versions.

FILE STRUCTURE

The following types are used:

LONG is a 32âbit integer stored in littleâendian format. *BYTE* is an 8âbit inte ger. *STRING* is a 0âterminated string of *BYTE*.

```
; Header

BYTE    ID[4]            ; "RGB9"
LONG    RevisionNumber   ; The format's revision number this file uses
LONG    NumberOfSymbols  ; The number of symbols used in this file
LONG    NumberOfSections ; The number of sections used in this file

; Symbols

REPT    NumberOfSymbols   ; Number of symbols defined in this object file.

    STRING  Name          ; The name of this symbol. Local symbols are stored
                          ; as "Scope.Symbol".

    BYTE    Type          ; 0 = LOCAL symbol only used in this file.
                          ; 1 = IMPORT this symbol from elsewhere
                          ; 2 = EXPORT this symbol to other objects.
                          ; Bit 7 is independent from the above value, and
                          ; encodes whether the section is unionized

    IF (Type & 0x7F) != 1 ; If symbol is defined in this object file.

        STRING  FileName  ; File where the symbol is defined.

        LONG    LineNum   ; Line number in the file where the symbol is defined.

        LONG    SectionID ; The section number (of this object file) in which
                          ; this symbol is defined. If it doesn't belong to any
                          ; specific section (like a constant), this field has
                          ; the value -1.

        LONG    Value     ; The symbols value. It's the offset into that
                          ; symbol's section.

    ENDC

ENDR
```

```
; Sections

REPT NumberOfSections
    STRING  Name  ; Name of the section

    LONG    Size  ; Size in bytes of this section

    BYTE    Type  ; 0 = WRAM0
                  ; 1 = VRAM
                  ; 2 = ROMX
                  ; 3 = ROM0
                  ; 4 = HRAM
                  ; 5 = WRAMX
                  ; 6 = SRAM
                  ; 7 = OAM

    LONG    Org   ; Address to fix this section at. -1 if the linker should
                  ; decide (floating address).

    LONG    Bank  ; Bank to load this section into. -1 if the linker should
                  ; decide (floating bank). This field is only valid for ROMX,
                  ; VRAM, WRAMX and SRAM sections.

    LONG    Align ; Alignment of this section, expressed as 1 << align. 1 if
                  ; not specified.

    IF      (Type == ROMX) || (Type == ROM0) ; Sections that can contain data.

        BYTE    Data[Size]      ; Raw data of the section.

        LONG    NumberOfPatches ; Number of patches to apply.

        ; These types of sections may have patches

        REPT    NumberOfPatches

            STRING  SourceFile   ; Name of the source file (for printing error
                                 ; messages).

            LONG    Offset       ; Offset into the section where patch should
                                 ; be applied (in bytes).

            BYTE    Type         ; 0 = BYTE patch.
                                 ; 1 = little endian WORD patch.
                                 ; 2 = little endian LONG patch.
                                 ; 3 = JR offset value BYTE patch.

            LONG    RPNSize      ; Size of the buffer with the RPN.
                                 ; expression.

            BYTE    RPN[RPNSize] ; RPN expression. Definition below.
```

```
        ENDR

    ENDC

ENDR

; Assertions

LONG  NumberOfAssertions

REPT  NumberOfAssertions

  STRING  SourceFile   ; Name of the source file (for printing the failure).

  LONG    Offset       ; Offset into the section where the assertion is located.

  BYTE    Type         ; 0 = Prints the message but allows linking to continue
                       ; 1 = Prints the message and evaluates other assertions,
                       ;     but linking fails afterwards
                       ; 2 = Prints the message and immediately fails linking

  LONG    RPNSize      ; Size of the RPN expression's buffer.

  BYTE    RPN[RPNSize] ; RPN expression, same as patches. Assert fails if == 0.

  LONG    SectionID    ; The section number (of this object file) in which this
                       ; assert is defined. If it doesn't belong to any specific
                       ; section (like a constant), this field has the value -1.

  STRING  Message      ; A message displayed when the assert fails. If set to
                       ; the empty string, a generic message is printed instead.

ENDR
```

RPN DATA

Expressions in the object file are stored as RPN. This is an expression of the form “2 5 +”. This will first push the value “2” to the stack, then “5”. The “+” operator pops two arguments from the stack, adds them, and then pushes the result on the stack, effectively replacing the two top arguments with their sum. In the RGB format, RPN expressions are stored as *BYTE*s with some bytes being special prefixes for integers and symbols.

Value	Meaning
$00	+ operator
$01	- operator
$02	* operator
$03	/ operator
$04	% operator
$05	unary -
$10	| operator
$11	& operator

`$12` `^` operator
`$13` unary `~`
`$21` `&&` comparison
`$22` `||` comparison
`$23` unary`!`
`$30` `==` comparison
`$31` `!=` comparison
`$32` `>` comparison
`$33` `<` comparison
`$34` `>=` comparison
`$35` `<=` comparison
`$40` `<<` operator
`$41` `>>` operator
`$50` `BANK(symbol)`, a *`LONG`* Symbol ID follows.
`$51` `BANK(section_name)`, a null-terminated string follows.
`$52` `Current BANK()`
`$60` `HRAMCheck`. Checks if the value is in HRAM, ANDs it with 0xFF.
`$61` `RSTCheck`. Checks if the value is a RST vector, ORs it with 0xC7.
`$80` *`LONG`* integer follows.
`$81` *`LONG`* symbol ID follows.

SEE ALSO

`rgbasm`(1), `rgblink`(1), `rgbds`(7), `gbz80`(7)

HISTORY

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