

# BLIT Command -- User Manual

## Overview

The BLIT commands provide fast block transfer of rectangular pixel regions. They are the primary mechanism for moving image data between buffers, loading images from files or flash, scaling, compositing layers, and drawing compressed or raw image data from memory.

## Buffer Identifiers

Several BLIT commands use single-letter buffer identifiers:

Buffer	Name	Description
F	Framebuffer	Primary off-screen buffer (created with FRAMEBUF)
L	Layer	Secondary overlay buffer (created with FRAMEBUF)
N	Display	Physical display buffer (VGA/HDMI only, requires R)
2	Second FB	Second framebuffer (VGA/HDMI only)
T	Top Layer	Additional overlay (RP2350 VGA/HDMI only)

## BLIT (bare)

Copies a rectangular region of pixels within the current write buffer.

## Syntax

```
BLIT x1, y1, x2, y2, w, h
```

## Parameters

Parameter	Description
x1, y1	Source top-left coordinates
x2, y2	Destination top-left coordinates
w	Width in pixels (must be $\geq 1$ )
h	Height in pixels (must be $\geq 1$ )

## Behaviour

- Copies from (x1, y1) to (x2, y2) within the current write buffer / display.
- Handles overlapping source and destination regions correctly.
- Coordinates are clipped to the screen bounds (HRes x VRes).
- Returns silently if  $w < 1$ ,  $h < 1$ , or the region is fully off-screen.

## Example

# BLIT Command -- User Manual

```
' Copy a 64x64 region from (0,0) to (100,50)
BLIT 0, 0, 100, 50, 64, 64
```

## BLIT READ

Captures a rectangular region from the screen into a numbered blit buffer.

### Syntax

```
BLIT READ #n, x, y, w, h
```

### Parameters

Parameter	Description
#n	Buffer number (1-64). The # prefix is optional.
x, y	Top-left corner of the screen region to capture
w	Width in pixels (must be $\geq 1$ )
h	Height in pixels (must be $\geq 1$ )

### Behaviour

- Reads the screen region into the buffer as 24-bit RGB data.
- The buffer must not already be in use. Use BLIT CLOSE to free a buffer before reusing it.
- Negative coordinates are clipped. Returns silently if  $w < 1$  or  $h < 1$ .

### Example

```
' Capture a 32x32 region from the screen
BLIT READ #1, 100, 50, 32, 32
```

## BLIT WRITE

Draws a previously-captured blit buffer to the screen.

### Syntax

```
BLIT WRITE #n, x, y [, mode]
```

### Parameters

Parameter	Description
-----------	-------------

# BLIT Command -- User Manual

#n	Buffer number (1-64)
x, y	Destination top-left coordinates
mode	Optional mode flags (0-7, default 0)

## Mode Flags

Bit	Value	Effect
0	1	Mirror horizontal
1	2	Mirror vertical
2	4	Transparent (black pixels are not drawn)

Flags can be combined. For example, mode 5 = mirror horizontal + transparent.

## Behaviour

- Draws the buffer contents at the specified position.
- Supports horizontal/vertical mirroring and transparency.
- With transparency (mode bit 2), black pixels (RGB 0,0,0) are skipped, allowing the background to show through.
- Handles partial off-screen clipping.
- The buffer must have been previously filled with BLIT READ or BLIT LOAD.

## Example

```
BLIT READ #1, 0, 0, 32, 32      ' Capture
BLIT WRITE #1, 200, 100         ' Draw at new position
BLIT WRITE #1, 200, 100, 5      ' Draw mirrored horizontally + transparent
```

## BLIT CLOSE

Frees a blit buffer and releases its memory.

## Syntax

```
BLIT CLOSE #n
```

## Parameters

Parameter	Description
#n	Buffer number (1-64)

## Example

```
BLIT READ #1, 0, 0, 32, 32
' ... use the buffer ...
```

# BLIT Command -- User Manual

```
BLIT CLOSE #1          ' Free the memory
```

---

## BLIT LOAD / BLIT LOADBMP

Loads a BMP image file (or a sub-region of it) into a blit buffer.

### Syntax

```
BLIT LOAD #n, filename$ [, x_offset, y_offset [, width, height]]
```

BLIT LOADBMP is accepted as an alternative spelling.

### Parameters

Parameter	Description
#n	Buffer number (1-64)
filename\$	Path to BMP file (.bmp extension auto-appended if missing)
x_offset	X origin within the BMP to start loading (default 0)
y_offset	Y origin within the BMP to start loading (default 0)
width	Width of sub-region to load (default: image width - x_offset)
height	Height of sub-region to load (default: image height - y_offset)

### Behaviour

- Loads the BMP as 24-bit RGB data into the buffer.
- The buffer must not already be in use.
- Requires an SD card to be present.
- The sub-region specified by offset and dimensions must fit within the BMP image.

### Example

```
' Load an entire BMP
BLIT LOAD #1, "sprite.bmp"

' Load a 32x32 sub-region starting at (64, 0)
BLIT LOAD #2, "spritesheet.bmp", 64, 0, 32, 32

BLIT WRITE #1, 100, 50          ' Draw to screen
```

---

## BLIT MERGE

Composites a rectangular region of the Layer buffer on top of the Framebuffer with transparency, and copies the result to the physical LCD display. This is the partial-area equivalent of FRAMEBUFFER MERGE. Both commands apply only

# BLIT Command -- User Manual

to LCD screens.

## Syntax

```
BLIT MERGE [colour], x, y, w, h [, mode [, timer]]
```

## Parameters

Parameter	Description
colour	RGB121 colour index (0-15) treated as transparent during the merge (default)
x, y	Top-left corner of the region to merge
w, h	Width and height of the merge region
mode	Optional: B (background), R (repeating), A (abort) -- SPI LCD builds only
timer	Repeat interval in milliseconds (0-600000, only with mode R)

## Mode Options (SPI LCD Builds Only)

Mode	Description
B	Run the merge once in the background (on core 1)
R	Run the merge repeatedly at the specified timer interval
A	Abort a running repeating merge

## Behaviour

- Only available on LCD builds (not VGA or HDMI). FRAMEBUFFER MERGE provides the full-screen equivalent.
- Requires both a Framebuffer and a Layer to exist.
- For each pixel in the specified region: if the Layer pixel matches colour, the Framebuffer pixel is used; otherwise the Layer pixel is used.
- The composited result is sent directly to the physical LCD -- neither the Framebuffer nor the Layer buffer is modified.
- The background and repeating modes allow the merge to run on core 1, freeing the main core for game logic.

VGA/HDMI note: On VGA and HDMI builds, BLIT MERGE and FRAMEBUFFER MERGE do not exist because layer compositing is performed automatically as part of the scanline output. The Layer buffer is composited on top of the display buffer every frame with no explicit command needed. On RP2350 in certain modes, a Top Layer (T) is also available, creating a three-layer composite: Top Layer over Layer over Display buffer.

## Example

```
' LCD display example
FRAMEBUFFER CREATE
FRAMEBUFFER LAYER

' Draw game world on framebuffer, HUD on layer
FRAMEBUFFER WRITE F
' ... draw world ...
FRAMEBUFFER WRITE L
CLS 0
TEXT 10, 10, "SCORE: 1000"
```

# BLIT Command -- User Manual

```
' Composite the HUD region onto the world and send to LCD
' (colour 0 = transparent, so only the text appears over the world)
BLIT MERGE 0, 0, 0, 320, 32
```

## BLIT RESIZE

Scales a rectangular source region into a destination rectangle using fast nearest-neighbour sampling. Operates on RGB121-packed buffers (4bpp).

### Syntax

```
BLIT RESIZE src, dst, sx, sy, sw, sh, dx, dy, dw, dh [, transparent]
```

### Parameters

Parameter	Description
src	Source buffer: L, F, N (VGA/HDMI), 2 (VGA/HDMI), T (RP2350 VGA/HDMI)
dst	Destination buffer: L, F, N (VGA/HDMI), 2 (VGA/HDMI), T (RP2350 VGA/HDMI)
sx, sy	Top-left source coordinate
sw, sh	Source width and height (must be > 0)
dx, dy	Top-left destination coordinate
dw, dh	Destination width and height (must be > 0)
transparent	Optional RGB121 colour index to skip (0-15). Default -1 (disabled).

### Behaviour

- Uses nearest-neighbour scaling (fast pixel replication/subsampling).
- Source rectangle validation is strict: must lie entirely within HRes x VRes.
- Destination rectangle is clipped to display bounds.
- If source and destination overlap in the same buffer, a temporary copy is automatically used.
- If transparent is 0-15, source pixels matching that colour are not written to the destination.
- If src or dst is N, the display must be in RGB121 mode (MODE 2 or MODE 3).

### Memory Use

- Non-overlap path: no temporary buffer allocated.
- Overlap path: temporary copy allocated as  $((sw + 1) \gg 1) * sh$  bytes.

### Example

```
' Scale a 64x64 region from framebuffer to layer as 128x96
BLIT RESIZE F, L, 32, 16, 64, 64, 100, 40, 128, 96

' Scale with transparency: skip source pixel value 0
```

# BLIT Command -- User Manual

```
BLIT RESIZE F, L, 32, 16, 64, 64, 100, 40, 128, 96, 0
```

## BLIT FLASH

Copies a rectangular region from a flash image slot into a framebuffer. Flash images are loaded once with FLASH LOAD IMAGE and persist in flash memory, allowing fast rendering without SD card access during gameplay.

### Syntax

```
BLIT FLASH slot, dst, x1, y1, x2, y2, w, h [, transparent]
```

### Parameters

Parameter	Description
slot	Flash image slot number (1-3)
dst	Destination buffer: L, F, N (VGA only), T (RP2350 VGA only)
x1, y1	Top-left corner within the flash image (source region origin)
x2, y2	Top-left corner on the destination buffer
w, h	Width and height to copy
transparent	Optional RGB121 colour (0-15) to treat as transparent. Default -1 (disabled)

### Behaviour

- The flash image must have been previously loaded with FLASH LOAD IMAGE.
- Images are stored in RGB121 format in flash.
- Source region must fit within the flash image dimensions.
- On non-VGA builds, N is not available as a destination.

### Example

```
' Load tileset into flash (one-time setup)
FLASH LOAD IMAGE 1, "background.bmp"

' During gameplay: blit from flash to framebuffer
BLIT FLASH 1, F, 0, 0, 0, 0, 320, 240

' Blit a 64x16 UI element with transparency
BLIT FLASH 2, F, 0, 0, 250, 5, 64, 16, 0
```

## BLIT FRAMEBUFFER

Copies a rectangular region between framebuffers (Layer, Framebuffer, Display, Top Layer).

# BLIT Command -- User Manual

## Syntax

```
BLIT FRAMEBUFFER src, dst, x1, y1, x2, y2, w, h [, transparent]
```

## Parameters

Parameter	Description
src	Source buffer: L, F, N, T (RP2350 VGA only)
dst	Destination buffer: L, F, N, T (RP2350 VGA only)
x1, y1	Source region top-left
x2, y2	Destination top-left
w, h	Width and height to copy
transparent	Optional RGB121 colour (0-15) to treat as transparent. Default -1 (disabled)

## Behaviour

- Source and destination must be different buffers.
- When both buffers are in-memory, uses fast RGB121 nibble copy.
- On non-VGA builds with LCD displays, N reads from or writes to the physical LCD panel.
- Supports transparency: pixels matching the transparent colour are not written.

## Example

```
' Copy a region from framebuffer to layer
BLIT FRAMEBUFFER F, L, 0, 0, 0, 0, 320, 240

' Copy with transparency
BLIT FRAMEBUFFER L, F, 10, 10, 50, 50, 100, 80, 0
```

---

## BLIT COMPRESSED

Draws RLE-compressed 4-bit image data from a memory address to the screen.

## Syntax

```
BLIT COMPRESSED address, x, y [, transparent]
```

## Parameters

Parameter	Description
address	Memory address of the compressed data
x, y	Destination top-left on screen
transparent	Optional RGB121 colour (0-15) to skip. Default -1 (disabled).



# BLIT Command -- User Manual

## Behaviour

- The data at address starts with a 4-byte header: two uint16\_t values for width and height.
- The image data following the header is RLE-compressed in 4-bit (nibble) format.
- Width must be <= HRes and height must be <= VRes.
- Typically used with data generated by CFunction tools or stored in program memory.

## Example

```
' Draw compressed image from a known memory address
BLIT COMPRESSED PEEK(CFUNADDR img_data), 50, 30
```

---

## BLIT MEMORY

Draws uncompressed (or auto-detected compressed) 4-bit image data from a memory address.

## Syntax

```
BLIT MEMORY address, x, y [, transparent]
```

## Parameters

Parameter	Description
address	Memory address of the image data
x, y	Destination top-left on screen
transparent	Optional RGB121 colour (0-15) to skip. Default -1 (disabled).

## Behaviour

- The data starts with a 4-byte header: two uint16\_t values for width and height.
- If the high bit (0x8000) of either dimension is set, the data is treated as RLE-compressed and decompressed automatically. The lower 15 bits give the actual dimensions.
- For uncompressed data, pixels are stored as packed 4-bit nibbles (2 pixels per byte).
- Supports transparency: pixels matching the transparent colour are not drawn.
- Uses optimised byte-level copy when alignment permits (x even, w even, no transparency).

## Example

```
' Draw nibble-packed image data from memory
BLIT MEMORY img_addr%, 0, 0

' Draw with transparency (colour 0 = transparent)
BLIT MEMORY img_addr%, 100, 50, 0
```

# BLIT Command -- User Manual

## Summary

Command	Purpose	All Builds?
BLIT x1,y1,x2,y2,w,h	Copy region within current buffer	Yes
BLIT READ #n,x,y,w,h	Capture screen region to buffer	Yes
BLIT WRITE #n,x,y[,mode]	Draw buffer to screen (mirror/transparent)	Yes
BLIT CLOSE #n	Free a blit buffer	Yes
BLIT LOAD #n,file\$[,xo,yo,w,h]	Load BMP into buffer	Yes (needs SD)
BLIT MERGE [c],x,y,w,h[,mode,t]	Composite layer over framebuffer to LCD	LCD only
BLIT RESIZE s,d,sx,sy,sw,sh,dx,dy,dw,dh[,t]	Scale/resize between buffers	Yes
BLIT FLASH slot,d,x1,y1,x2,y2,w,h[,t]	Draw from flash image slot	Yes
BLIT FRAMEBUFFER s,d,x1,y1,x2,y2,w,h[,t]	Copy between framebuffers	Yes
BLIT COMPRESSED addr,x,y[,t]	Draw RLE-compressed nibble data	Yes
BLIT MEMORY addr,x,y[,t]	Draw raw/compressed nibble data	Yes

Up to 64 blit buffers (numbered 1-64) can be allocated simultaneously. Flash image slots (1-3) are separate from blit buffers.