

SPRITE Command and Function Reference Manual

Overview

The SPRITE system provides firmware-accelerated sprite graphics for the PicoMite. Sprites are rectangular graphic objects that can be moved independently over a background, with automatic collision detection and layer management.

Key Features:

- Up to 64 sprites (numbered 1-64)
- Up to 5 layers (0-4) for z-ordering
- Automatic sprite-to-sprite collision detection
- Edge-of-screen collision detection
- Static object collision detection
- Sprite rotation and mirroring
- Efficient batch movement operations
- Background scrolling with wrap-around

SPRITE Commands

Loading and Creating Sprites

SPRITE LOAD

Loads sprites from a sprite definition file.

```
SPRITE LOAD filename$ [, start_sprite] [, mode]
```

Parameter	Description
filename\$	Path to the sprite file (.spr extension added if omitted)
start_sprite	First sprite buffer to load into (1-64, default: 1)
mode	Color palette mode: 0 = standard, 1 = alternate (default: 0)

Sprite File Format: The first line contains: width, count [, height]. If height is omitted, sprites are assumed square. Each sprite is defined as height lines of width characters. Characters 0-9 and A-F represent colors (hex values). Space represents transparent pixels. Lines starting with apostrophe are comments.

SPRITE LOADARRAY

Loads a sprite from a numeric array.

```
SPRITE LOADARRAY #n, width, height, array()
```

Parameter	Description
#n	Sprite buffer number (1-64)
width	Sprite width in pixels
height	Sprite height in pixels
array()	Numeric array containing color values

SPRITE LOADPNG (RP2350 only)

Loads a sprite from a PNG file.

```
SPRITE LOADPNG #n, filename$ [, transparent] [, cutoff]
```

Parameter	Description
#n	Sprite buffer number (1-64)
filename\$	Path to PNG file (.png extension added if omitted)
transparent	Transparent color index (0-15, default: 0)
cutoff	Alpha threshold for transparency (1-254, default: 30)

SPRITE Command and Function Reference Manual

SPRITE LOADBMP

Loads a sprite from a BMP file.

```
SPRITE LOADBMP #n, filename$ [, x_offset, y_offset, width, height]
```

Parameter	Description
#n	Sprite buffer number (1-64)
filename\$	Path to BMP file (.bmp extension added if omitted)
x_offset	X offset within image (default: 0)
y_offset	Y offset within image (default: 0)
width	Width to load (default: full image)
height	Height to load (default: full image)

SPRITE READ

Reads a sprite directly from the screen.

```
SPRITE READ #n, x, y, width, height
```

Parameter	Description
#n	Sprite buffer number (1-64)
x, y	Top-left corner of screen area to read
width, height	Size of area to capture

SPRITE COPY

Creates copies of a sprite that share the same image data.

```
SPRITE COPY #source, #dest, count
```

Parameter	Description
#source	Source sprite buffer number
#dest	First destination sprite buffer number
count	Number of copies to create

Notes: Copies share the same image data (memory efficient). Each copy has its own position and state. Cannot copy a copy. Must close all copies before closing the source.

SPRITE Command and Function Reference Manual

Displaying Sprites

SPRITE SHOW

Displays a sprite at a specified position.

```
SPRITE SHOW #n, x, y, layer [, rotation]
```

Parameter	Description
#n	Sprite buffer number (1-64)
x, y	Display position (can extend off-screen)
layer	Display layer (0-4)
rotation	Rotation/mirror value (0-7, default: 0)

Rotation Values:

Value	Effect
0	Normal
1	Mirror horizontal
2	Mirror vertical
3	Mirror both (180 degree rotation)
4-7	Same as 0-3 but with transparency disabled

Layer Behavior: Layer 0 scrolls with background (SPRITE SCROLL). Layers 1-4 are fixed position, rendered in order.

Example:

```
SPRITE SHOW #1, 100, 50, 1, 0
```

SPRITE SHOW SAFE

Displays a sprite safely, properly handling overlapping sprites.

```
SPRITE SHOW SAFE #n, x, y, layer [, rotation] [, newbuffer]
```

Use when moving sprites that may overlap with other sprites. Automatically hides and redraws overlapping sprites. Slower than SPRITE SHOW but produces correct results.

SPRITE WRITE

Draws a sprite directly to the screen without tracking.

```
SPRITE WRITE #n, x, y [, rotation]
```

Does not store background or track position. Does not participate in collision detection. Use for static decorative elements.

Hiding Sprites

SPRITE HIDE

Hides a single sprite.

```
SPRITE HIDE #n
```

Restores the background where the sprite was displayed. Sprite remains loaded and can be shown again.

SPRITE HIDE SAFE

Safely hides a sprite, handling overlapping sprites.

```
SPRITE HIDE SAFE #n
```

Properly redraws overlapping sprites. Slower than SPRITE HIDE but produces correct results.

SPRITE Command and Function Reference Manual

SPRITE HIDE ALL

Hides all currently displayed sprites.

```
SPRITE HIDE ALL
```

Sprites remain loaded and can be restored. Use before drawing to the background.

SPRITE RESTORE

Restores all sprites hidden by SPRITE HIDE ALL.

```
SPRITE RESTORE
```

SPRITE Command and Function Reference Manual

Moving Sprites

SPRITE NEXT

Sets the next position for a sprite (used with SPRITE MOVE).

```
SPRITE NEXT #n, x, y
```

Does not immediately move the sprite. Position is applied when SPRITE MOVE is called.

SPRITE MOVE

Executes all pending SPRITE NEXT movements.

```
SPRITE MOVE
```

Moves all sprites to their NEXT positions in one operation. More efficient than individual SPRITE SHOW commands.

SPRITE SWAP

Swaps a displayed sprite with a hidden sprite of the same size.

```
SPRITE SWAP #displayed, #hidden [, rotation]
```

Efficient sprite animation technique. Both sprites must be the same size.

Scrolling

SPRITE SCROLL

Scrolls the background and layer 0 sprites.

```
SPRITE SCROLL x, y [, fill_color]
```

Parameter	Description
x	Horizontal scroll amount (positive = right)
y	Vertical scroll amount (positive = down)
fill_color	Color for exposed areas, -1 no fill, -2 wrap-around (default)

Layer 0 sprites scroll with the background. Layers 1-4 remain stationary. Static objects also scroll.

Closing Sprites

SPRITE CLOSE

Closes a single sprite and frees its memory.

```
SPRITE CLOSE #n
```

Hides the sprite if displayed. Cannot close a sprite that has active copies.

SPRITE CLOSE ALL

Closes all sprites and static objects.

```
SPRITE CLOSE ALL
```

Collision Detection

SPRITE INTERRUPT

Sets the interrupt handler for sprite-to-sprite and edge collisions.

```
SPRITE INTERRUPT label
```

Called when a sprite collides with another sprite or screen edge.

SPRITE NOINTERRUPT

Disables the sprite collision interrupt.

```
SPRITE NOINTERRUPT
```

SPRITE Command and Function Reference Manual

Static Objects

Static objects are invisible rectangular regions that trigger collisions when sprites intersect them. Useful for walls, platforms, obstacles, and trigger zones.

SPRITE STATIC

Defines or removes a static object.

```
SPRITE STATIC #n, x, y, width, height ' Define
SPRITE STATIC #n, OFF ' Remove
```

Parameter	Description
#n	Static object number (1-64)
x, y	Position of the object
width, height	Size of the object

SPRITE STATIC CLEAR

Removes all static objects.

```
SPRITE STATIC CLEAR
```

SPRITE STINTERRUPT

Sets the interrupt handler for static object collisions.

```
SPRITE STINTERRUPT label
```

Called when a sprite collides with a static object. Use SPRITE(ST, COLLISION) and SPRITE(ST, OBJECT) to get details.

SPRITE NOSTINTERRUPT

Disables the static object collision interrupt.

```
SPRITE NOSTINTERRUPT
```

Miscellaneous

SPRITE SET TRANSPARENT

Sets the transparent color for all sprites.

```
SPRITE SET TRANSPARENT color
```

SPRITE Command and Function Reference Manual

SPRITE() Function

The SPRITE() function returns information about sprites, collisions, and background objects.

Sprite Properties

SPRITE(W, #n)

Returns the width of sprite #n in pixels. Returns -1 if sprite not loaded.

SPRITE(H, #n)

Returns the height of sprite #n in pixels. Returns -1 if sprite not loaded.

SPRITE(X, #n)

Returns the X position of sprite #n. Returns 10000 if sprite not displayed.

SPRITE(Y, #n)

Returns the Y position of sprite #n. Returns 10000 if sprite not displayed.

SPRITE(L, #n)

Returns the layer of sprite #n. Returns -1 if sprite not displayed.

SPRITE(A, #n)

Returns the memory address of sprite #n's image data.

Collision Information

SPRITE(C, #n [, index])

Returns collision information for sprite #n.

Without index: Returns the number of collisions detected.

With index: Returns the sprite number or edge code for that collision.

Edge Collision Codes:

Code	Meaning
0xF1	Left edge
0xF2	Top edge
0xF4	Right edge
0xF8	Bottom edge
0x80-0xBF	Static object (object number = code AND 0x3F)

SPRITE(T, #n)

Returns the cumulative collision bitmask for sprite #n. Each bit represents a sprite that has collided.

SPRITE(E, #n)

Returns the edge collision flags for sprite #n. Bit 1=Left, 2=Top, 4=Right, 8=Bottom.

SPRITE(S)

Returns the sprite number that triggered the last collision interrupt.

Distance and Direction

SPRITE(V, #n1, #n2)

Returns the angle (in radians) from sprite #n1 to sprite #n2, measured clockwise from north.

SPRITE Command and Function Reference Manual

SPRITE(D, #n1, #n2)

Returns the distance in pixels between the centers of sprites #n1 and #n2.

Background Collision Detection

SPRITE(B, #n)

Performs pixel-level collision detection between the sprite and the background it is covering. This function compares the sprite's non-transparent pixels against the background stored when the sprite was displayed.

Returns:

Value	Description
0	No collision - sprite does not overlap non-transparent background
1	Background overlap - sprite covers non-transparent background pixels
2	Pixel collision - sprite non-transparent pixels overlap background

Notes: The sprite must be active (displayed) for this function to work. Uses the blitstoreptr buffer which contains the background captured when the sprite was shown. Updates the backgroundcollision[] array with detailed collision information.

SPRITE(B, #n, side)

Returns detailed collision penetration information after calling SPRITE(B, #n).

Bounding Box Collision (sides 0-3):

Side	Description
0	Right-most collision X offset from sprite left edge
1	Left-most collision X offset from sprite right edge
2	Bottom-most collision Y offset from sprite top edge
3	Top-most collision Y offset from sprite bottom edge

Pixel-Level Collision (sides 4-7):

Side	Description
4	Penetration depth from sprite left bound
5	Penetration depth from sprite right bound
6	Penetration depth from sprite top bound
7	Penetration depth from sprite bottom bound

Notes: Only one of values 4 or 5 will be non-zero (indicates horizontal collision side). Only one of values 6 or 7 will be non-zero (indicates vertical collision side). Use these values to determine collision response direction (push-back).

Count Information

SPRITE(N)

Returns the total number of sprites currently displayed.

SPRITE(N, layer)

Returns the number of sprites on the specified layer (0-4).

SPRITE Command and Function Reference Manual

Static Object Properties

SPRITE(ST, #n, X)

Returns the X position of static object #n. Returns -1 if not defined.

SPRITE(ST, #n, Y)

Returns the Y position of static object #n. Returns -1 if not defined.

SPRITE(ST, #n, W)

Returns the width of static object #n. Returns -1 if not defined.

SPRITE(ST, #n, H)

Returns the height of static object #n. Returns -1 if not defined.

SPRITE(ST, #n, A)

Returns 1 if static object #n is active, 0 otherwise.

SPRITE(ST, COLLISION)

Returns the sprite number that collided with a static object (set when STINTERRUPT fires).

SPRITE(ST, OBJECT)

Returns the static object number that was hit (set when STINTERRUPT fires).

SPRITE Command and Function Reference Manual

Memory Usage

Understanding how sprites use memory helps you plan your program and avoid "Not enough Heap memory" errors.

How Much Memory Do Sprites Use?

Each sprite uses heap memory for three purposes:

- Image data: ~256 bytes - The actual pixels of the sprite
- Background buffer: (included above) - Saves what's behind the sprite
- Bounds data: ~256 bytes - Used for pixel-perfect collision detection

Rule of thumb: Each sprite uses approximately 500-600 bytes of heap memory, regardless of the sprite's pixel dimensions (for small to medium sprites up to about 20x20 pixels).

Memory Usage Examples

- 10 sprites: ~5-6 KB
- 32 sprites: ~16-19 KB
- 64 sprites: ~35-40 KB

Tips for Managing Memory

1. Check available memory before creating sprites:

Print "Heap free: "; MM.INFO(HEAP FREE)

2. Close sprites you no longer need:

SPRITE CLOSE n ' Close a specific sprite

SPRITE CLOSE ALL ' Close all sprites and free their memory

3. Use SPRITE COPY for multiple identical sprites:

When you need many copies of the same sprite image (like bullets or particles), copies share the original's image data, saving significant memory.

4. Smaller sprites use the same minimum memory:

Due to memory allocation in 256-byte blocks, a 4x4 sprite uses the same memory as a 16x16 sprite. Consider this when designing your graphics.

5. Leave headroom for your program:

Your BASIC program, arrays, and strings also use heap memory. Don't allocate all available memory to sprites.

Checking Memory During Development

```
Dim integer start_mem = MM.INFO(HEAP FREE)
Print "Before sprites: "; start_mem

' ... create your sprites ...

Print "After sprites: "; MM.INFO(HEAP FREE)
Print "Sprites used: "; start_mem - MM.INFO(HEAP FREE); " bytes"
```

SPRITE Command and Function Reference Manual

Technical Notes

Performance Tips

1. Use SPRITE SHOW for non-overlapping sprites
2. Use SPRITE SHOW SAFE when sprites may overlap
3. Batch movements with SPRITE NEXT and SPRITE MOVE
4. Use SPRITE SWAP for animation instead of loading new images
5. Use SPRITE HIDE ALL before drawing to the background
6. Minimize the number of active layers

Limitations

- Maximum 64 sprite buffers
- Maximum 64 static objects
- Maximum 5 layers (0-4)
- Maximum 4 simultaneous collision reports per sprite
- Sprites must fit within screen resolution
- Only available on VGA and HDMI displays or framebuffers. Not available on SPI, I2C or parallel connected displays

SPRITE Command and Function Reference Manual

Example Programs

Basic Sprite Display

```
' Load and display a sprite
SPRITE LOAD "player.spr", 1
SPRITE SHOW #1, 160, 120, 1
```

Collision Detection

```
SPRITE LOAD "player.spr", 1
SPRITE LOAD "enemy.spr", 2

SPRITE INTERRUPT collision_handler
SPRITE SHOW #1, 100, 100, 1
SPRITE SHOW #2, 150, 100, 1

DO
    ' Game logic here
LOOP

collision_handler:
    which = SPRITE(S)
    PRINT "Sprite"; which; "collided!"
    IRETURN
```

Static Objects for Walls

```
' Create a simple room with walls
SPRITE STATIC #1, 0, 0, 320, 10      ' Top wall
SPRITE STATIC #2, 0, 230, 320, 10    ' Bottom wall
SPRITE STATIC #3, 0, 0, 10, 240      ' Left wall
SPRITE STATIC #4, 310, 0, 10, 240    ' Right wall

SPRITE STINTERRUPT wall_hit
SPRITE LOAD "player.spr", 1
SPRITE SHOW #1, 160, 120, 1

' ... game code ...

wall_hit:
    PRINT "Hit wall"; SPRITE(ST, OBJECT)
    IRETURN
```

Smooth Animation with SPRITE NEXT/MOVE

```
SPRITE LOAD "anim.spr", 1, 0
FOR i = 2 TO 10
    SPRITE COPY #1, #i, 1
NEXT i

' Display all sprites
FOR i = 1 TO 10
    SPRITE SHOW #i, i * 30, 100, 1
NEXT i

' Animate
DO
    FOR i = 1 TO 10
        x = SPRITE(X, #i)
        y = SPRITE(Y, #i)
        SPRITE NEXT #i, x + 1, y + SIN(TIMER/100 + i) * 2
    NEXT i
    SPRITE MOVE
    PAUSE 16
```

SPRITE Command and Function Reference Manual

LOOP