TILEMAP Command -- User Manual

Overview

The TILEMAP command provides hardware-accelerated tile map rendering for 2D games on RGB121 displays. It combines flash-resident tilesets (loaded via FLASH LOAD IMAGE) with compact internal map storage read from DATA statements to efficiently render scrolling tile-based worlds.

Map data uses just 2 bytes per cell (instead of 8 bytes per MMBasic integer), so a 100×50 map occupies only 10 KB internally. Tile attributes can be defined separately, allowing collision detection to distinguish solid, climbable, collectible and other tile types.

Instead of blitting each tile individually from BASIC (which would require hundreds of BLIT FLASH calls per frame), TILEMAP DRAW renders all visible tiles in a single C-level command with automatic sub-tile smooth scrolling and viewport clipping.

Prerequisites

- An RGB121 framebuffer -- either a VGA/HDMI display in RGB121 mode, or an LCD display with FRAMEBUFFER CREATE (use FRAMEBUFFER COPY to update the physical display)
- A tileset image loaded into flash via FLASH LOAD IMAGE
- Map data defined in DATA statements

Tileset Image Format

The tileset is a BMP image containing all tiles arranged in a grid. Load it into one of the 3 flash slots:

```
FLASH LOAD IMAGE 1, "tileset.bmp"
```

The tileset image should have tiles arranged in a regular grid. For example, a 256×64 pixel image with 16×16 pixel tiles contains 16 tiles across and 4 rows = 64 tiles total.

Tile indices are 1-based: tile 1 is the top-left tile in the image, tile 2 is the next one to the right, and so on, wrapping to the next row.

Tile index 0 is reserved as "empty" -- these cells are skipped during rendering.

Commands

TILEMAP CREATE mapLabel, id, flashSlot, tileW, tileH, tilesPerRow, cols, rows

Creates a tilemap by reading map data from DATA statements into compact internal storage.

Parameter	Range	Description
mapLabel	label	Label before the DATA statements containing the m
id	1-4	Tilemap slot number (up to 4 simultaneous tilemaps
flashSlot	1-3	Flash image slot containing the tileset
tileW	1-256	Width of each tile in pixels
tileH	1-256	Height of each tile in pixels

tilesPerRow	1-1024	Number of tiles across the tileset image
cols	1-10000	Number of columns in the map
rows	1-10000	Number of rows in the map

The DATA statements must contain at least cols * rows integer values (0-65535), read row by row. Tile index 0 = empty, 1+ = tile from the tileset.

The map is stored internally as uint16_t (2 bytes per cell). The user's READ/RESTORE position is not affected.

Example:

```
FLASH LOAD IMAGE 1, "tiles.bmp"
TILEMAP CREATE mapdata, 1, 1, 16, 16, 16, 10, 3
END

mapdata:
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
DATA 0, 0, 2, 0, 0, 0, 3, 0, 0, 0
DATA 5, 5, 5, 5, 5, 5, 5, 5, 5, 5
```

TILEMAP ATTR attrLabel, id, numTiles

Associates a tile attribute table with a tilemap. Attributes are properties of tile types in the tileset image (e.g. solid, climbable, damaging), not specific map positions. Multiple tilemaps sharing the same tileset would each need their own ATTR call.

Parameter	Range	Description
attrLabel	label	Label before the DATA statements containing attrib
id	1-4	Tilemap slot number
numTiles	1-65535	Number of attribute entries to read

The DATA must contain at least numTiles integer values. Entry 1 corresponds to tile 1, entry 2 to tile 2, etc. Each value is a bitmask of user-defined flags.

Example:

```
' Define attribute bits
CONST SOLID = &b0001
CONST LADDER = &b0010
CONST DAMAGE = &b0100
CONST COLLECT = &b1000

TILEMAP ATTR tileattrs, 1, 6
END

tileattrs:
DATA 0          ' tile 1: empty/passable
DATA SOLID      ' tile 2: brick (solid)
DATA SOLID      ' tile 3: stone (solid)
DATA LADDER     ' tile 4: ladder
DATA COLLECT    ' tile 5: coin (collectible)
DATA DAMAGE     ' tile 6: spikes (damaging)
```

TILEMAP Command -- User Manual

TILEMAP DESTROY id

Destroys a tilemap and frees its internal memory (map and attribute table).

```
TILEMAP DESTROY 1
```

TILEMAP SET id, col, row, tileIndex

Sets a single tile in the internal map. Useful for interactive changes (removing a coin when collected, opening a door, breaking a block).

Parameter	Description
id	Tilemap slot number
col	Column (0-based)
row	Row (0-based)
tileIndex	New tile index (0-65535, 0 = empty)

Example:

```
TILEMAP SET 1, 12, 5, 0       ' Remove the tile at column 12, row 5
```

TILEMAP DRAW id, dest, viewX, viewY, screenX, screenY, viewW, viewH [, transparent]

Renders all visible tiles to a destination buffer. This is the core rendering command.

Parameter	Description
id	Tilemap slot number
dest	Destination buffer: L (Layer), F (Framebuffer), N (Display), T (Top layer - R
viewX, viewY	Pixel offset into the world (viewport top-left corner, supports sub-tile precis
screenX, screenY	Where on the destination buffer to start drawing
viewW, viewH	Size of the visible viewport in pixels
transparent	Optional: RGB121 colour index (0-15) to treat as transparent within tiles. -1

The viewport coordinates (viewX, viewY) provide sub-tile smooth scrolling. Partially visible tiles at edges are automatically clipped.

Tiles with map index 0 are always skipped (treated as empty space). The optional transparent parameter defines which colour within non-zero tiles should be treated as see-through -- useful for irregularly shaped tiles.

Example:

```
' Draw the visible portion of the map to the framebuffer
TILEMAP DRAW 1, F, camX, camY, 0, 0, 320, 240

' Same, but with colour 0 (BLACK) as transparent within tiles
TILEMAP DRAW 1, F, camX, camY, 0, 0, 320, 240, 0
```

TILEMAP SCROLL id, dx, dy

Adjusts the stored viewport position by a relative offset and clamps to world bounds. Use this for simple camera control without tracking the position yourself.

Parameter	Description
id	Tilemap slot number
dx, dy	Pixel offset to add to the current viewport position

The viewport is clamped so it doesn't scroll past the edges of the map. After calling SCROLL, use TILEMAP(VIEWX id) and TILEMAP(VIEWY id) to read the resulting position.

Note: TILEMAP SCROLL only updates the stored viewport -- it does not render. Call TILEMAP DRAW afterwards to display the result.

```
TILEMAP SCROLL 1, 2, 0        ' Scroll right by 2 pixels
TILEMAP DRAW 1, F, TILEMAP(VIEWX 1), TILEMAP(VIEWY 1), 0, 0, 320, 240
```

TILEMAP VIEW id, x, y

Sets the viewport position to an absolute world-pixel coordinate.

```
TILEMAP VIEW 1, 0, 0          ' Reset to top-left
TILEMAP VIEW 1, playerX - 160, playerY - 120  ' Centre on player
```

TILEMAP CLOSE

Destroys all tilemaps (all 4 slots) and all sprites (all 64 slots), and frees all internal memory. Called automatically on program end.

```
TILEMAP CLOSE
```

Sprite Commands

The TILEMAP subsystem includes a lightweight sprite layer for game entities (player characters, enemies, projectiles, etc.). Up to 64 sprites can be active simultaneously. Each sprite references a tilemap's tileset for its graphics and is positioned at arbitrary pixel coordinates.

Sprites are rendered in one batch call (TILEMAP SPRITE DRAW) which iterates all 64 slots in C for maximum performance. Tile index 0 means "no tile" and is skipped.

TILEMAP SPRITE CREATE id, tilemapRef, tileIndex, x, y

Creates a sprite.

Parameter	Range	Description
id	1-64	Sprite slot number
tilemapRef	1-4	Which tilemap's tileset to use for graphics
tileIndex	1-65535	Initial tile to display (from the referenced tileset)
x, y	any integer	Initial screen position in pixels

```
' Create hero sprite using tilemap 1's tileset, tile 7, at centre of screen
TILEMAP SPRITE CREATE 1, 1, 7, 160, 120

' Create enemy using same tileset, tile 4
```

```
TILEMAP SPRITE CREATE 2, 1, 4, 40, 40
```

TILEMAP SPRITE MOVE id, x, y

Moves a sprite to a new pixel position.

```
TILEMAP SPRITE MOVE 1, playerX, playerY
```

TILEMAP SPRITE SET id, tileIndex

Changes the tile displayed by a sprite. Use this for animation (e.g. walking frames, blinking, direction changes).

```
' Toggle between two animation frames
IF frame MOD 10 < 5 THEN
  TILEMAP SPRITE SET 1, 7   ' frame A
ELSE
  TILEMAP SPRITE SET 1, 8   ' frame B
END IF
```

TILEMAP SPRITE DRAW dest, transparent

Renders all active sprites (up to 64) in one call. Sprites are drawn in slot order (1 first, 64 last), so higher-numbered sprites appear on top.

Parameter	Description
dest	Destination: F (framebuffer), N (display), or L (layer)
transparent	Colour index 0-15 treated as transparent, or -1 for no transparency

```
' Draw background tilemap first, then sprites on top
TILEMAP DRAW 1, F, camX, camY, 0, 0, 320, 240
TILEMAP SPRITE DRAW F, 0
```

TILEMAP SPRITE DESTROY id

Destroys a single sprite, clearing its slot.

```
TILEMAP SPRITE DESTROY 2
```

TILEMAP SPRITE CLOSE

Destroys all 64 sprites. Also called automatically by TILEMAP CLOSE.

```
TILEMAP SPRITE CLOSE
```

Functions

TILEMAP(TILE id, pixelX, pixelY)

Returns the tile index at the given world pixel coordinates. Returns 0 if the coordinates are outside the map bounds.

```
t = TILEMAP(TILE 1, playerX + 8, playerY + 16)
IF t = 5 THEN PRINT "Standing on a coin!"
```

TILEMAP(COLLISION id, x, y, w, h [, mask])

Tests whether a rectangular region (in world pixel coordinates) overlaps any non-zero tile. Returns the tile index of the first matching tile found, or 0 if the entire region is clear.

When called without mask, matches any non-zero tile (same as before). When mask is provided, only tiles whose attribute bits match are considered a hit -- the tile's attribute is ANDed with mask and the tile is only returned if the result is non-zero. Requires TILEMAP ATTR to have been called.

```
' Check if player hits any tile
hit = TILEMAP(COLLISION 1, px, py, 16, 24)
IF hit > 0 THEN PRINT "Collision with tile"; hit

' Check only solid tiles (using attribute mask)
CONST SOLID = &b0001
hit = TILEMAP(COLLISION 1, px, py, 16, 24, SOLID)
IF hit > 0 THEN PRINT "Blocked by solid tile"; hit
```

TILEMAP(ATTR id, tileIndex)

Returns the attribute flags for a given tile type. Returns 0 if no attribute table is loaded or the tile index is out of range.

```
a = TILEMAP(ATTR 1, 3)
IF a AND SOLID THEN PRINT "Tile 3 is solid"
IF a AND LADDER THEN PRINT "Tile 3 is a ladder"
```

TILEMAP(VIEWX id) / TILEMAP(VIEWY id)

Returns the current viewport X or Y position (set by DRAW, SCROLL, or VIEW).

```
PRINT "Camera at"; TILEMAP(VIEWX 1); ","; TILEMAP(VIEWY 1)
```

TILEMAP(COLS id) / TILEMAP(ROWS id)

Returns the number of columns or rows in the map.

```
PRINT "Map size:"; TILEMAP(COLS 1); "x"; TILEMAP(ROWS 1)
```

TILEMAP(SPRITE X id) / TILEMAP(SPRITE Y id)

Returns the current X or Y pixel position of a sprite.

```
PRINT "Hero at"; TILEMAP(SPRITE X 1); ","; TILEMAP(SPRITE Y 1)
```

TILEMAP(SPRITE TILE id)

Returns the current tile index displayed by a sprite.

```
IF TILEMAP(SPRITE TILE 2) = 4 THEN PRINT "Enemy showing coin tile"
```

TILEMAP Command -- User Manual

TILEMAP(SPRITE W id) / TILEMAP(SPRITE H id)

Returns the tile width or height (in pixels) of a sprite, inherited from its parent tilemap's tile dimensions.

```
PRINT "Sprite size:"; TILEMAP(SPRITE W 1); "x"; TILEMAP(SPRITE H 1)
```

TILEMAP(SPRITE HIT id1, id2)

Tests whether two sprites overlap using axis-aligned bounding box collision. Returns 1 if the sprites overlap, 0 otherwise. Each sprite's bounding box is determined by its position and its parent tilemap's tile dimensions.

```
FOR i = 2 TO 5
  IF TILEMAP(SPRITE HIT 1, i) THEN
    PRINT "Player hit enemy"; i
  END IF
NEXT i
```

Usage Patterns

Double-Buffered Game Loop

```
FRAMEBUFFER CREATE
DO
  FRAMEBUFFER WRITE F
  CLS RGB(COBALT)
  TILEMAP DRAW 1, F, camX, camY, 0, 0, 320, 240
  ' Draw sprites on top...
  FRAMEBUFFER COPY F, N
LOOP
```

Parallax Scrolling (Two Layers)

Use two tilemaps with different scroll rates:

```
TILEMAP CREATE bgdata, 1, 1, 16, 16, 8, 50, 15
TILEMAP CREATE fgdata, 2, 2, 16, 16, 16, 100, 15

' In game loop:
TILEMAP DRAW 1, F, camX\2, camY\2, 0, 0, 320, 240      ' BG at half speed
TILEMAP DRAW 2, F, camX, camY, 0, 0, 320, 240, 0        ' FG with transparency
```

Attribute-Based Collision

```
CONST SOLID = &b0001
CONST LADDER = &b0010
CONST COLLECT = &b1000

' Try moving right - only blocked by solid tiles
newX = px + speed
IF TILEMAP(COLLISION 1, newX, py, 14, 22, SOLID) = 0 THEN
  px = newX       ' No solid obstacle - allow movement
```

```
END IF

' Check for collectibles at player centre
t = TILEMAP(TILE 1, px + 8, py + 12)
IF t > 0 AND (TILEMAP(ATTR 1, t) AND COLLECT) THEN
  TILEMAP SET 1, (px + 8) \ 16, (py + 12) \ 16, 0   ' Remove collectible
  score = score + 100
END IF

' Check for ladder
IF TILEMAP(COLLISION 1, px + 4, py + 20, 8, 4, LADDER) > 0 THEN
  canClimb = 1
END IF
```

Gravity with Attribute Filtering

```
CONST SOLID = &b0001

' Try moving down (gravity)
newY = py + velY
IF TILEMAP(COLLISION 1, px, newY, 14, 22, SOLID) = 0 THEN
  py = newY       ' Falling
ELSE
  velY = 0        ' Landed
  py = (newY \ 16) * 16 - 22   ' Snap to tile boundary
END IF
```

Memory Usage

Map data is stored internally as uint16_t (2 bytes per cell):

Map Size	Internal Memory
100 x 15 (1500 cells)	3,000 bytes
200 x 30 (6000 cells)	12,000 bytes
500 x 50 (25000 cells)	50,000 bytes

Tile attributes add 2 bytes per tile type (e.g. 64 tile types = 128 bytes).

DATA statements reside in program memory and do not consume additional heap.

Limits

Resource	Limit
Simultaneous tilemaps	4
Simultaneous sprites	64
Flash image slots	3
Tile size	1-256 pixels per dimension
Tile index range	0-65535

Map dimensions	Up to 10000 x 10000 (limited by available RAM)
Attribute entries	Up to 65535
Tiles per tileset	Limited by flash image size

Performance Notes

- TILEMAP DRAW renders only the tiles visible within the viewport -- a 320x240 viewport with 16x16 tiles renders at most 21x16 = 336 tiles per frame, regardless of total map size.
- Each tile is rendered via the optimised blit121() function which uses fast aligned memcpy when possible.
- For best performance, use tile widths that are even numbers (2, 4, 8, 16, 32) to maximise aligned blit paths.
- Changes via TILEMAP SET take effect on the next DRAW call.
- The TILEMAP(COLLISION id, x, y, w, h, mask) attribute filter adds negligible overhead -- just one table lookup and AND per tile tested.