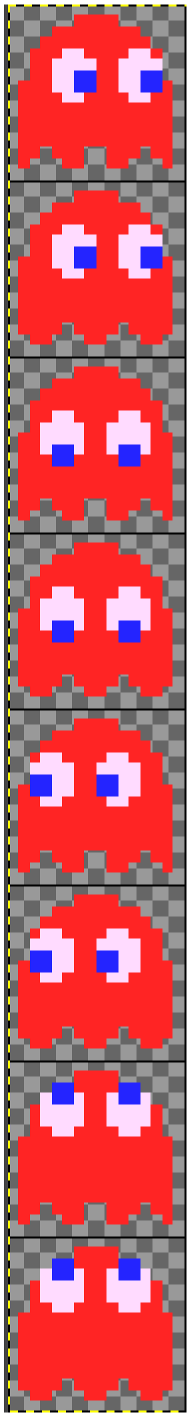
PAWSv2 a Risc-V (RV32IMAFCB or RV64GC+) CPU and SOC
DISPLAY – Sprite Layer

Sprite Layer

There are 2 sprite layers in PAWSv2, referred to as UPPER and LOWER. The UPPER layer sprites displayed above the LOWER sprites, but in all other respects the 2 layers offer identical functionality.

Each sprite layer contains 16 sprites, numbered 0 to 15, with sprite 15 showing above sprite 14, etc.

Sprite Details



A sprite is a 16x16 pixel tile, with 8 tiles per sprite, using 255 colours, plus transparent. A sprite tilestrip from the PACMAN game is shown to the left, where the grey chequerboard squares represent transparent, as shown in GIMP.

Sprite Parameters

Name	**Value Range**	**Description**
active	TRUE / FALSE	Display sprite flag
x	-1024 to 1023	Left display coordinate
y	-512 to 511	Top display coordinate
tile	0 to 7	Number of the tile to display
actions	See table below	Actions to apply to the sprite tile

The following defines are made in PAWSdefinitions.h to assist with generating the sprite actions, which can be OR'd together to achieve the required result. Note reflection and rotation are mutually exclusive.

Name	**Description**
SPRITE_DOUBLE	Display sprite with double sized pixels
SPRITE_QUAD	Display sprite with quadruple sized pixels
REFLECT_X	Reflect the tile in the x-axis
REFLECT_Y	Reflect the tile in the y-axis
ROTATE0	No rotation
ROTATE90	Rotate tile 90 degrees
ROTATE180	Rotate tile 180 degrees
ROTATE270	Rotate tile 270 degrees

The following PAWSlibrary function is used to set the initial display parameters of a sprite.

Function	**Description**
void set_sprite(unsigned char sprite_layer, unsigned char sprite_number, unsigned char active, short x, short y, unsigned char tile, unsigned char sprite_actions)	Set a sprite number sprite_number (0-15), in sprite_layer (UPPER/LOWER) to active, x, y, tile number (0-7) with sprite_actions.

A sprite can report all of its parameters, plus a within layer sprite-to-sprite collision flag, and a sprite-to-layer collision flag.

PAWSv2 a Risc-V (RV32IMAFCB or RV64GC+) CPU and SOC
DISPLAY – Sprite Layer

Other PAWSlibrary functions for sprites.

Function	Description
void set_sprite_bitmaps(unsigned char sprite_layer, unsigned char sprite_number, unsigned char *sprite_bitmaps)	Requires an array of 8x16x16 (2048) pixels to set an individual sprite tilesheet for sprite_layer (UPPER/LOWER), and sprite_number (0-15)
void set_sprite_bitamps_from_spritesheet(unsigned char sprite_layer, unsigned char *sprite_bitmaps)	Requires an array of 16x8x16x16 pixels to set the sprite tilesheet for ALL sprites in sprite_layer (UPPER/LOWER)
void set_sprite_attribute(unsigned char sprite_layer, unsigned char sprite_number, unsigned char attribute, short value)	Set an individual parameter for sprite_number (0-15) in sprite_layer (UPPER/LOWER) to value. Attribute is SPRITE_ACTIVE, SPRITE_TILE, SPRITE_X, SPRITE_Y, SPRITE_ACTION
short get_sprite_attribute(unsigned char sprite_layer, unsigned char sprite_number, unsigned char attribute)	Get an individual parameter for sprite_number (0-15) in sprite_layer (UPPER/LOWER) to value. Attribute is SPRITE_ACTIVE, SPRITE_TILE, SPRITE_X, SPRITE_Y, SPRITE_ACTION
unsigned short get_sprite_collision(unsigned char sprite_layer, unsigned char sprite_number)	
unsigned short get_sprite_layer_collision(unsigned char sprite_layer, unsigned char sprite_number)	Get the in-layer sprite-to-sprite collision flag for sprite_number (0-15) in sprite_layer (UPPER/LOWER). Returns a 16-bit field that shows when sprite_number is in collision with another sprite in the layer. NB: A sprite will always collide with itself if any pixels are displayed in that sprite.
void update_sprite(unsigned char sprite_layer, unsigned char sprite_number, unsigned short update_flag)	Get the sprite-to-layer collision flag for sprite_number (0-15) in sprite_layer (UPPER/LOWER). Returns a 4-bit field that shows when sprite_number is in collision with other layers. AND with SPRITE_TO_BITMAP, SPRITE_TO_LOWER_TILEMAP, SPRITE_TO_UPPER_TILEMAP or SPRITE_TO_OTHER_SPRITES to determine which layers the sprite is in collision with.

void update_sprite(unsigned char sprite_layer, unsigned char sprite_number, unsigned short update_flag)	Update sprite parameters for sprite_number (0-15) in layer (UPPER/LOWER).
	
	The update_flag is a 16-bit field as below:
	
	struct sprite_update_flag {
	    unsigned int padding:3;
	    unsigned int y_act:1;
	    unsigned int x_act:1;
	    unsigned int tile_act:1;
	    int dy:5;
	    int dx:5;
	};
	
	y_act and x_act, disable sprite if it goes off the screen, otherwise wrap. tile_act increment tile number for animation, dy and dx amount to move the sprite.

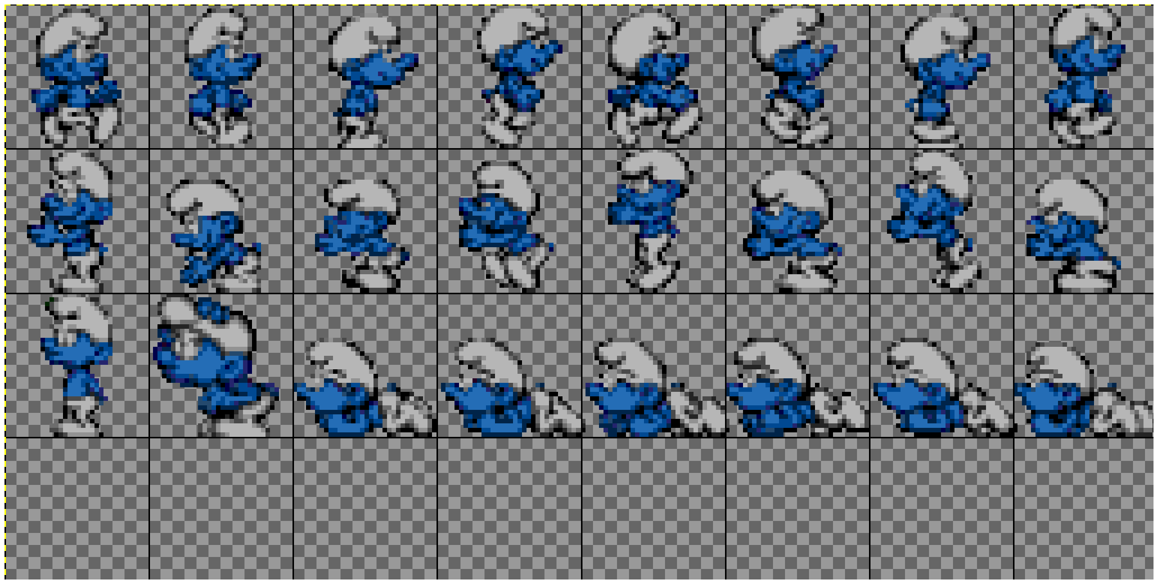
See ASTROIDS which uses sprites, sprite-to-sprite collisions, and the sprite-update system. See INVADERS which uses sprites, sprite-to-layer collisions, and the sprite-update system.

PAWSlibrary 32x32 Sprites

The PAWSlibrary allows the joining of 4 sprites to make a larger 32x32 sprite, and includes functions to simplify the setting and displaying of such sprites.

Function	Description
`void set_sprite_bitamps_from_spritesheet32x32( unsigned char sprite_layer, unsigned char *sprite_bitmaps )`	Requires an array of 4x8x32x32 pixels to set the sprite tilesheet for ALL sprites in sprite_layer (UPPER/LOWER)
`void set_sprite32( unsigned char sprite_layer, unsigned char sprite_number, unsigned char active, short x, short y, unsigned char tile, unsigned char sprite_actions )`	Set a sprite group sprite_number (0, 4, 8 or 12), in sprite_layer (UPPER/LOWER) to active, x, y, tile number (0-7) with sprite_actions. X and y are the CENTRE coordinates of the sprite group.

A 32x32 sprite tilesheet ready for conversion to C-style include format. The top line is for sprites 0-3 (tiles 0-7), second line sprites 4-7, third line 8-11, and the bottom line sprites 12-15.



See SMURFS which uses 32x32 sprites as a demonstration, including animation and reflection.

Utilities

In the Resources folder there is a GIMP image file PAWSv2.xcf that contains the PAWSv2 palette that can be imported into GIMP. Spritesheets for sprites and tilemaps can be created in GIMP, exported as GIF89 format, and then converted to C-style include files using the dumpsprite C program in the utility folder.