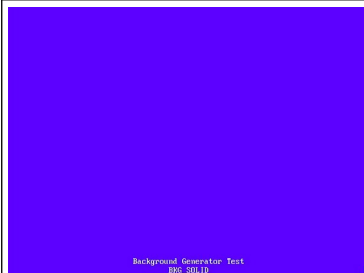
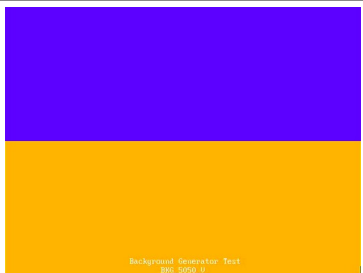
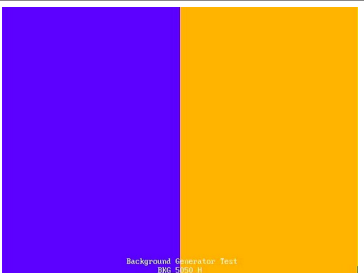
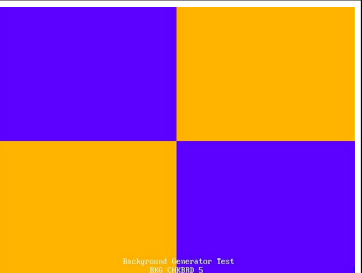
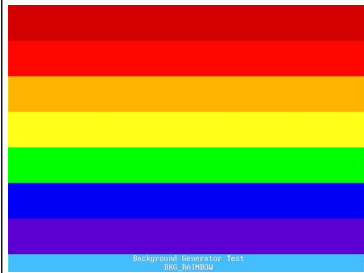
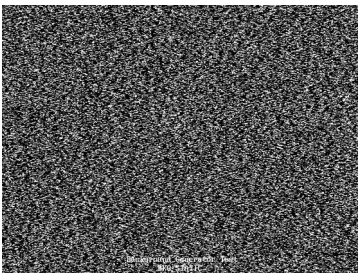


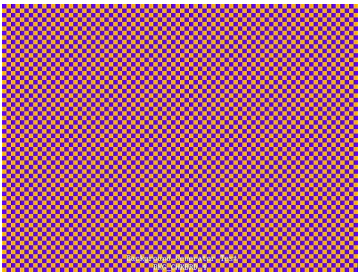
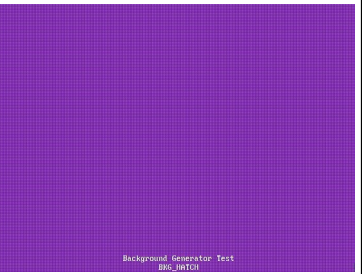
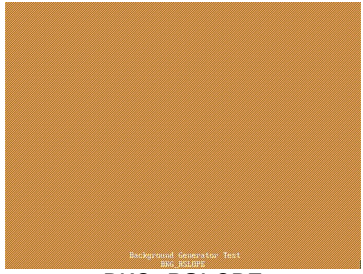

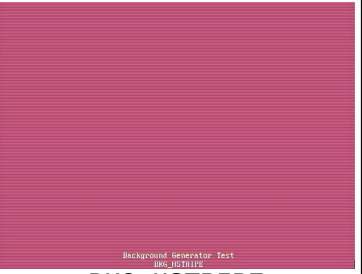


PAWSv2 a Risc-V (RV32IMAFCB or RV64GC+) CPU and SOC
Usage & Programming Guide

Background Generator

The background layer shows when there is nothing to display from the layers above. There are 16 named background generators in libPAWS. The result of the individual background generators are shown in the table below.

Result of set_background(PURPLE, ORANGE, value from table);			
 Background Generator Test BKG_SOLID	 Background Generator Test BKG_5050_V	 Background Generator Test BKG_5050_H	 Background Generator Test BKG_CHKBRD_5
 Background Generator Test BKG_RAINBOW	 Background Generator Test BKG_SNOW	 Background Generator Test BKG_STATIC	 Background Generator Test BKG_CHKBRD_1
 Background Generator Test BKG_CHKBRD_2	 Background Generator Test BKG_CHKBRD_3	 Background Generator Test BKG_CHKBRD_4	 Background Generator Test BKG_HATCH
 Background Generator Test BKG_LSLOPE	 Background Generator Test BKG_RSLOPE	 Background Generator Test BKG_VSTRIPE	 Background Generator Test BKG_HSTRIPE

In addition, a simple co-processor, called COPPER, is available to change background generator parameters during the frame generation.

PAWSv2 a Risc-V (RV32IMAFCB or RV64GC+) CPU and SOC

Usage & Programming Guide

Background COPPER Programming (New Copper NuCu)

The COPPER is designed to allow the changing of the background generator parameters during the display frame., referred to as NuCU.

NuCu has 4 registers for data storage, and 4 psuedo registers; when reading giving access to the pixel coordinates and vblank status; when writing setting the background generator parameters; plus a CPU read/write value accessible.

READ		WRITE	
CU_RB	VBLANK status	CU_BM	Background MODE
CU_RX	X pixel coordinate	CU_BC	Background COLOUR
CU_RY	Y pixel coordinate	CU_BA	Background ALT COLOUR
CU_RC	CPU input value	CU_RC	CPU output value
CU_R0	Register 0	CU_R0	Register 0
CU_R1	Register 1	CU_R1	Register 1
CU_R2	Register 2	CU_R2	Register 2
CU_R3	Register 3	CU_R3	Register 3
In the instruction table below, REG1 refers to both the destination register AND the first operand for two register operations. (X) would refer to the value of the X register.			

NuCu has a fixed instruction set, with 4 fields as detailed below.

OPCODE	DEST REG AND REG1	REG2 REG / LITERAL FLAG	REG2 / LITERAL	EXAMPLE	MEANING
CU_JMP		Y	Y	JMP 7	JUMP to RL
CU_SET	Y	Y	Y	SET R1 \leftarrow (R0)	SET reg1 to RL
CU_ADD	Y	Y	Y	ADD R0 \leftarrow (R0) + 1	ADD RL to (reg1) store in reg1
CU_SUB	Y	Y	Y	SUB R0 \leftarrow (R0) - 1	SUB RL from (reg1) store in reg1
CU_AND	Y	Y	Y	AND R0 \leftarrow (R0) & 7	AND (reg1) by RL store in reg1
CU_OR	Y	Y	Y	OR R0 \leftarrow (R0) 1	OR (reg1) by RL store in reg1
CU_XOR	Y	Y	Y	XOR R0 \leftarrow (R0) ^ 1	XOR (reg1) by RL store in reg1
CU_SHL	Y	Y	Y	SHL (R0) \leftarrow (R0) << 6	Shift Left (reg1) by RL store in reg1
CU_SHR	Y	Y	Y	SHR (R0) \leftarrow (R0) >> 1	Shift Right (reg1) by RL store in reg1
CU_SEQ	Y	Y	Y	SEQ (R0) == (Y)	Skip next if (reg1) == RL
CU_SNE	Y	Y	Y	SNE (R0) == (X)	Skip next if (reg1) != RL
CU_SLT	Y	Y	Y	SLT (R0) < (R1)	Skip next if (reg1) < RL
CU_SLE	Y	Y	Y	SLE (R0) <= (R1)	Skip next if (reg1) <= RL
CU_RND	Y	Y	Y	SET R0 \leftarrow RAND & R1	SET reg1 to RANDOM & RL
CU_LFM	Y	Y	Y	LFM R0 \leftarrow [(R1)]	LOAD [RL] store in reg1
CU_STM	Y	Y	Y	STM [(R1)] \leftarrow RL	STORE RL in [(reg1)]
(reg) refers to value of the register. [addr] refers directly to contents of a memory cell. [(reg)] refers to the cotents of a memory cell identified by the value of the register. RL refers to (reg2) or literal value depending on the register/literal flag. SKIP instructions act like CHIP8 and bypass the following instruction if the condition is true.					

PAWSv2 a Risc-V (RV32IMAFCB or RV64GC+) CPU and SOC

Usage & Programming Guide

NuCu has 8 memory locations, which can be set by the CPU before the NuCu program starts running. These can be read by the LFM instruction, and set by the STM instruction in the NuCu program.

NuCu has program storage for 128 entries.

A simple NUCU program that sets the background generator to the BKG_SNOW pattern on a BLACK background, and changes the colour of the snow/stars every 64 pixels down the screen to give a rainbow effect.

```
copper_startstop( 0 );

unsigned short memoryinit[8] = {
    WHITE,
    RED,
    ORANGE,
    YELLOW,
    GREEN,
    LTBLUE,
    PURPLE,
    MAGENTA
};

copper_set_memory( memoryinit );           // PROGRAM COPPER MEMORY ARRAY OF COLOURS

copper_program( 0, CU_SET, CU_BM, CU_RL, BKG_SNOW ); // BACKGROUND SNOW GENERATOR
copper_program( 1, CU_SET, CU_BA, CU_RL, BLACK );    // BACKGROUND ALT BLACK
copper_program( 2, CU_SET, CU_BC, CU_RL, WHITE );    // BACKGROUND WHITE

copper_program( 3, CU_SET, CU_R0, CU_RL, 0 );        // SET R0 = 0

copper_program( 4, CU_SET, CU_R1, CU_RR, CU_R0 );    // SET R1 = R0
copper_program( 5, CU_SHL, CU_R1, CU_RL, 6 );        // R1 = R1 * 64
copper_program( 6, CU_LFM, CU_R2, CU_RR, CU_R0 );    // R2 = MEM[ (R0) ]

copper_program( 7, CU_SEQ, CU_RY, CU_RR, CU_R1 );    // Y == R1 ?
copper_program( 8, CU_JMP, FALSE, CU_RL, 7 );        // SKIP YES, ELSE GO TO 7

copper_program( 9, CU_SET, CU_BC, CU_RR, CU_R2 );    // SET BACKGROUND = R2
copper_program( 10, CU_ADD, CU_R0, CU_RL, 1 );       // R0 = R0 + 1
copper_program( 11, CU_AND, CU_R0, CU_RL, 7 );       // R0 = R0 & 7
copper_program( 12, CU_JMP, FALSE, CU_RL, 4 );       // JUMP 4

copper_startstop( 1 );
```

See ASTROIDS / INVADERS / DEMO for this NuCu program running.