JEAT board specification and manual

rev. 1.1

SPI interface specification

Sending data to the JEAT board is straight-forward, but with a twist or two. CS to the board needs to go from high to low before the first byte is sent. The messages are put together like this:

	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
Byte 0	N/A	N/A	N/A	N/A	A19	A18	A17	A16
Byte 1	A15	A14	A13	A12	A11	A10	A9	A8
Byte 2	A7	A6	A5	A4	A3	A2	A1	A0
Byte 3	DWZ	N/A	N/A	N/A	N/A	N/A	N/A	WRAP 8
Byte 4	WRAP 7	WRAP 6	WRAP 5	WRAP 4	WRAP 3	WRAP 2	WRAP 1	WRAP 0
Byte 5	D0_7	D0_6	D0_5	D0_4	D0_3	D0_2	D0_1	D0_0
Byte 6	D1_7	D1_6	D1_5	D1_4	D1_3	D1_2	D1_1	D1_0
...								

A19-A0 The address you want to start writing to.

WRAP8-0 This is a 9-bit value letting you update only a square in a bitmap using only one transfer. If you want to write to a 8x8 pixel area, simply set the address to the top left pixel address, set wrap to 8 and let your 64 bytes data follow. Wrap = 0 → Wrap is not used.

DWZ Don't write zeroes. If you are writing to a bitmap but don't want to overwrite the pixels you write to with transparent pixels, set DWZ to 1 and this will be taken care of. Pure magic.

D0 First data byte

D1 Second data byte

So, just set the address (A19-A0) to the adress you want to start writing to, figure out if you want to use DWZ or not and what Wrap value to use. Then send the ”head” (Byte 0-4) and just let your data follow. You can write the whole memory in one transfer, if you like.

For every sent byte a byte is returned. This byte tells how much is left in the sound buffer. The returned value is only 8 bits, multiplying it by 8 gives an idea of how many bytes are left in the 2k buffer.

Address	Bitmap registers							
0x00000 - 0x7FFFF	Video RAM							
0x80000	L1X_7	L1X_6	L1X_5	L1X_4	L1X_3	L1X_2	L1X_1	L1X_0
0x80001	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L1X_8
0x80002	L1Y_7	L1Y_6	L1Y_5	L1Y_4	L1Y_3	L1Y_2	L1Y_1	L1Y_0
0x80003	L2X_7	L2X_6	L2X_5	L2X_4	L2X_3	L2X_2	L2X_1	L2X_0
0x80004	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L2X_8
0x80005	L2Y_7	L2Y_6	L2Y_5	L2Y_4	L2Y_3	L2Y_2	L2Y_1	L2Y_0
0x80006	L3X_7	L3X_6	L3X_5	L3X_4	L3X_3	L3X_2	L3X_1	L3X_0
0x80007	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L3X_8
0x80008	L3Y_7	L3Y_6	L3Y_5	L3Y_4	L3Y_3	L3Y_2	L3Y_1	L3Y_0
0x80009	N/A	N/A	N/A	N/A	TestImg	L3_Act	L2_Act	L1_Act
0x8000A	Background color. If nothing else is visible on a pixel, this color is shown.							
0x80010	L1XTL_7	L1XTL_6	L1XTL_5	L1XTL_4	L1XTL_3	L1XTL_2	L1XTL_1	L1XTL_0
0x80011	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L1XTL_8
0x80012	L1YTL_7	L1YTL_6	L1YTL_5	L1YTL_4	L1YTL_3	L1YTL_2	L1YTL_1	L1YTL_0
0x80013	L1XBR_7	L1XBR_6	L1XBR_5	L1XBR_4	L1XBR_3	L1XBR_2	L1XBR_1	L1XBR_0
0x80014	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L1XBR_8
0x80015	L1YBR_7	L1YBR_6	L1YBR_5	L1YBR_4	L1YBR_3	L1YBR_2	L1YBR_1	L1YBR_0
0x80018	L2XTL_7	L2XTL_6	L2XTL_5	L2XTL_4	L2XTL_3	L2XTL_2	L2XTL_1	L2XTL_0
0x80019	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L2XTL_8
0x8001A	L2YTL_7	L2YTL_6	L2YTL_5	L2YTL_4	L2YTL_3	L2YTL_2	L2YTL_1	L2YTL_0
0x8001B	L2XBR_7	L2XBR_6	L2XBR_5	L2XBR_4	L2XBR_3	L2XBR_2	L2XBR_1	L2XBR_0
0x8001C	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L2XBR_8
0x8001D	L2YBR_7	L2YBR_6	L2YBR_5	L2YBR_4	L2YBR_3	L2YBR_2	L2YBR_1	L2YBR_0

Address	Bitmap registers							
0x80020	L3XTL_7	L3XTL_6	L3XTL_5	L3XTL_4	L3XTL_3	L3XTL_2	L3XTL_1	L3XTL_0
0x80021	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L3XTL_8
0x80022	L3YTL_7	L3YTL_6	L3YTL_5	L3YTL_4	L3YTL_3	L3YTL_2	L3YTL_1	L3YTL_0
0x80023	L3XBR_7	L3XBR_6	L3XBR_5	L3XBR_4	L3XBR_3	L3XBR_2	L3XBR_1	L3XBR_0
0x80024	N/A	N/A	N/A	N/A	N/A	N/A	N/A	L3XBR_8
0x80025	L3YBR_7	L3YBR_6	L3YBR_5	L3YBR_4	L3YBR_3	L3YBR_2	L3YBR_1	L3YBR_0

LxX Layer x X position offset

LxY Layer x Y position offset

LxyTL Layer x will NOT be visible above or to the left of this pixel position (top-left)

LxyBR Layer x will NOT be visible below or to the right of this pixel position (bottom-right)

TestImg If this is set to 1, the video output will show a test image.

Lx_Act Used to tell what bitmaps should be visible.

Sprite blocks

There are 27 sprites available. Each of them has a block describing how they should behave. The block of sprite 0 is at 0x81000, sprite 1 is at 0x81010 and so on.

Offset	Sprite block							
0x00	XPos_7	XPos_6	XPos_5	XPos_4	XPos_3	XPos_2	XPos_1	XPos_0
0x01	Visible	N/A	N/A	N/A	N/A	N/A	XPos_8	YPos_8
0x02	YPos_7	YPos_6	YPos_5	YPos_4	YPos_3	YPos_2	YPos_1	YPos_0
0x03	Sprite shape. Shape 0 – 127 is in sprite-only RAM, 128 – 255 is in bitmap 3 RAM area.							
0x04	Sprite layer. 0 – in front of bitmap 0, 1 – behind bitmap 0 and in front of bitmap 1, 2 – behind bitmap 1 and in front of bitmap 2, 3 – behind bitmap 2. Lower sprite number has priority if sprites is seen on the same pixel and being on the same layer.							
0x08	SxXTL_7	SxXTL_6	SxXTL_5	SxXTL_4	SxXTL_3	SxXTL_2	SxXTL_1	SxXTL_0
0x09	N/A	N/A	N/A	N/A	N/A	N/A	N/A	SxXTL_8
0x0A	SxYTL_7	SxYTL_6	SxYTL_5	SxYTL_4	SxYTL_3	SxYTL_2	SxYTL_1	SxYTL_0
0x0B	SxXBR_7	SxXBR_6	SxXBR_5	SxXBR_4	SxXBR_3	SxXBR_2	SxXBR_1	SxXBR_0
0x0C	N/A	N/A	N/A	N/A	N/A	N/A	N/A	SxXBR_8
0x0D	SxYBR_7	SxYBR_6	SxYBR_5	SxYBR_4	SxYBR_3	SxYBR_2	SxYBR_1	SxYBR_0

XPos, YPos The position of the top left pixel of the sprite.

Visible 1 – Sprite is visible, 0 – sprite is not visible

Sprite shape This is where the graphics data of the sprite is stored. A sprite always uses 1k (1024 bytes) of memory. Sprite shape 0 is at RAM adress 0x00000, shape 1 is at 0x00400 and so on. 128 sprite shapes are always available, and 128 more can be used if bitmap 3 is not used at the same time.

Sprite layer Tells where the sprite should be positioned, described in the table.

SxyTL, SxyBR The sprite is only visible in the window described by these two pixels (Top Left and Bottom Right).

Registers 0x00-0x02 should be written in address order. When 0x02 has been written values in 0x00-0x02 is clocked in and being used after the next vertical sync. This is to prevent strange effects if position data is written during a vertical sync.

Video RAM memory map and bitmaps

The RAM is divided in four sections; Bitmap 0, 1, 2 / Spriteshapes and Spriteshapes. The bitmap RAM sections are 128k each, allowing 512x256 pixels with 8 bit each. With bitmap offsets X and Y set to zero, the first byte in the bitmap memory will show on the top left pixel. The next byte will show to the right of the first pixel. The first pixel on the second pixel line will reside in byte 513 (adress 512). Since not the whole bitmap memory is shown on-screen you can freely update bitmap contents off-screen and scroll it in by using the offset.

The second section can be used either as bitmap 2 or as sprite shapes.

Adress	Section
0x00000 - 0x1FFFF	Sprite shapes 0-127
0x20000 - 0x3FFFF	Bitmap 2 or sprite shapes 128-255
0x40000 - 0x5FFFF	Bitmap 1
0x60000 - 0x7FFFF	Bitmap 0

Color

The board can show 255 colors on screen at the same time, plus transparancy (color nr 0). The 1k color RAM is used as a look-up table. Color 0 is always transparent, but all other colors can be set to any 24-bit RGB color. The color RAM starts at address 0x82000.

Address	Color Nr	A + 0	A + 1	A + 2	A + 3
0x82000	0	N/A	N/A	N/A	N/A
0x82004	1	Blue	Green	Red	N/A
0x82008	2	Blue	Green	Red	N/A
0x8200C	3	Blue	Green	Red	N/A
...	...	...	...	...	...

Sound

The on-board sound uses an 8-bit PWM output from the FPGA. The sample speed is currently fixed to 44100Hz. The output is filtered and fed to the on-board amplifier.

Playing sound is simple. For every byte you send to the board (using SPI) a byte is returned. Multiply this byte by 8 and you know how many 8-bit samples are left in the 2k buffer. Filling the buffer at each vertical sync is enough.

Writing to any adress 0x88xxx puts the written data in the sound buffer in the order it is written, no matter the address.

It should also be noted that you don't have to use the FPGA sound. The 3.5mm connector on the board can be used as input or output if you want to use another amplifier or use the on-board amplifier with another sound generator.

Pinout

The pinout of the controller connector has been designed to be directly connected to a Raspberry Pi. However, you can use any controller that has an SPI interface and enough I/O for the controls you want to use.

Pin	Description
1	This should be connected to the voltage you want to use with your player controller I/O's, normally 3.3 or 5V. This is connected to the pullup resistors of the JAMMA controller inputs.
2	5V power supply, connected to the JAMMA 5V pins.
3	Connected to JAMMA TEST.
4	Not connected.
5	Connected to JAMMA COIN pin.
6	0V
7	Connected to JAMMA Player 2 Start.
8	Connected to JAMMA Player 1 Start.
9	0V
10	Connected to JAMMA Player 2 Up.
11	Connected to JAMMA Player 1 Up.
12	Connected to JAMMA Player 2 Down.
13	Connected to JAMMA Player 1 Down.
14	0V
15	Connected to JAMMA Player 2 Left.
16	Connected to JAMMA Player 1 Left.
17	Not connected.
18	Connected to JAMMA Player 2 Right.
19	SPI MOSI (3.3V signal level)
20	0V
21	SPI MISO (3.3V signal level)
22	VSync. This signal changes state at each vertical sync. In other words, a 30Hz square wave syncronized with the vertical sync.
23	SPI SCLK (3.3V signal level)
24	SPI /CS (3.3V signal level)
25	0V
26	Not connected.
27	Not connected.
28	Not connected.
29	Connected to JAMMA Player 1 Button 4.
30	0V
31	Connected to JAMMA Player 2 Button 4.
32	Connected to JAMMA Player 1 Button 3.
33	Connected to JAMMA Player 2 Button 3.
34	0V
35	Connected to JAMMA Player 1 Button 2.

Pin	Description
36	Connected to JAMMA Player 2 Button 2.
37	Connected to JAMMA Player 1 Button 1.
38	Connected to JAMMA Player 2 Button 1.
39	0V
40	Connected to JAMMA Player 1 Right.