JeST_006

'PSX JoyPad Adapter for Atari digital interfaces'

March 3rd 2009

Author : techie_alison

www.atari-forum.com

MANUAL

INTRODUCTION

There is a requirement among the users of retro computers to control their equipment with a joystick. These joysticks are becoming rarer today, and the modern games console joypads are far superior and more comfortable to use.

Traditional joysticks are little more than a collection of switches, where as the modern joypads communicate via a high speed clocked serial link. A means of interfacing the two is required. This is relatively straightforward by means of a multi I/O microcontroller.

Said microcontroller will interface with the serial bus on one side, and the other side will emulate said switches. The complicated bit is deciding what happens in the middle. What buttons will do what? What extra features can we build into such a device?

This document will bring together all of the information required at hand to develop this project. The first release will realise the interface in it's most basic form, which simply comprises of up, down, left, right, and the fire buttons. Thereafter additional functionality will be explored.

At the time of writing the current paragraph, 3rd March 2009, the interface has been completed and sits at revision _006. It's features are as follows :-

- Atari style joystick emulation with thumbpad
- Atari ST mouse emulation with thumbstick and thumbpad
- Thumbstick buttons operate as mouse buttons
- Variable thumbstick mouse speed
- Fully assignable fire buttons
- Fully assignable autofire buttons
- Fully assignable auto leftright buttons (e.g. HyperSports)
- Autofire repeat rate can be changed
- FIRE1 mapped to pin-6 (left button)
- FIRE2 mapped to Pin-9 (right button)
- FIRE3 mapped to Pin-5 (middle button, 7800 FIRE2)
- Amiga 3-button mouse emulation
- Tested with Atari ST, Amiga, Commodore-64, Sinclair Spectrum
- Compatible with Atari 7800 (uses pin-5)
- Fully user configurable
- Four savable user setups
- Original 'factory' setups recoverable
- Draws 30mA 5v in operation
- Extremely small and tidy, it couldn't be smaller

1.0 ENTERING COMMANDS

Commands fall into two groups, those without a variable, and those with. We shall call them *regular* and *variable* commands. A regular command is entered by pressing **SELECT** and keeping it held firmly throughout, this tells JeST to listen. While you are holding **SELECT**, you then press the various buttons on the controller one by one. While you are doing this a tally is kept of each button pressed, such that the required buttons can be entered in any order.

When issuing a regular command, two counters are filled with binary data. That data is then passed to a *command-handler* when **SELECT** is finally released. Unrecognised commands are simply not trapped and are thus ignored.

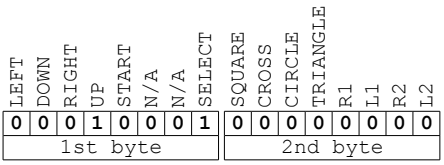


Fig. How JeST records commands

The command above is **LOAD_SETUP_1**. It does not expect a variable. The sequence would be described as **SELECT+UP**.

Commands with variables are almost the same except that **START** is pressed once when entering the command. To issue **PROGRAM_FIRE_BUTTON_1** you would press **SELECT+SQUARE+START**. Having done this, you then need to enter in the variable. There is no button to hold down to make JeST recognise it, as you had already pressed **START** when issuing the command.

To enter the variable, you simply press the various buttons to tally up the binary number, and then to action it you press **START** at the very end. The command (with it's variable) is now passed to the command-handler.

The following sequence will assign fire button 1 to every single button on your controller. **SELECT+SQUARE+START, SQUARE+CROSS+CIRCLE+TRIANGLE+R1+L1+R2+L2, START**. This is what it looks like to JeST.

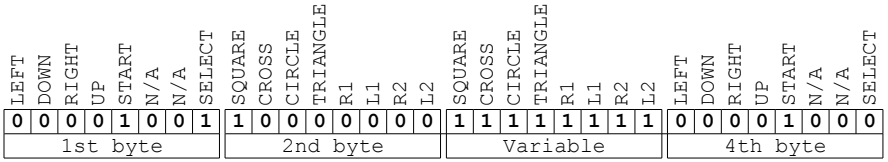


Fig. Assign all PSX buttons to FIRE1

If you accidentally press START while issuing a command, then JeST will appear to have stopped. What it is actually doing is waiting for you to enter variable data, which will be followed by **START** to action it. Just press **START** to leave the entry mode.

Commands or variables can be entered in any order in their tally block, **SQUARE+CIRCLE+L1** is identical to **CIRCLE+L1+SQUARE**. From JeST's perspective it simply sees a binary number of **10100100**.

SELECT must always be held while entering the command, and **START** must be pressed once at the very end to action a variable, ...if one was required.

1.1 COMMANDS - FRIENDLY NAMES

Friendly Name : Load Setup 1
Outline : Loads a setup which is stored in memory.
Entry : **SELECT+UP**
Variable Expected : No

Friendly Name : Load Setup 2
Outline : Loads a setup which is stored in memory.
Entry : **SELECT+RIGHT**
Variable Expected : No

Friendly Name : Load Setup 3
Outline : Loads a setup which is stored in memory.
Entry : **SELECT+DOWN**
Variable Expected : No

Friendly Name : Load Setup 4
Outline : Loads a setup which is stored in memory.
Entry : **SELECT+LEFT**
Variable Expected : No

Friendly Name : Save Setup 1
Outline : Saves the current values in use to the target setup memory slot.
Entry : **SELECT+UP+L2**
Variable Expected : No

Friendly Name : Save Setup 2
Outline : Saves the current values in use to the target setup memory slot.
Entry : **SELECT+RIGHT+L2**
Variable Expected : No

Friendly Name : Save Setup 3
Outline : Saves the current values in use to the target setup memory slot.
Entry : **SELECT+DOWN+L2**
Variable Expected : No

Friendly Name : Save Setup 4
Outline : Saves the current values in use to the target setup memory slot.
Entry : **SELECT+LEFT+L2**
Variable Expected : No

Friendly Name : Reset Entire Setup Memory
Outline : Fills all of the setup slots with 'factory' data.
Entry : **SELECT+L1+L2+R1+R2+SQUARE+CROSS+TRIANGLE+CIRCLE+START, L1+L2+R1+R2+SQUARE+CROSS+TRIANGLE+CIRCLE, START**
Variable Expected : Yes

Friendly Name : Set Auto Repeat Rate
Outline : Changes the timing of the repeat rate. A higher number means a longer delay. Suggested values are between 00000001 (1) and 00010000 (32). A delay of 5 will yield a rate of about 6 times per second. Affects both AUTOFIRE and AUTOLEFTRIGHT.
Entry : **SELECT+SQUARE+CROSS+CIRCLE+TRIANGLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign FIRE 1
Outline : Records PSX button presses which will be assigned to Atari FIRE1.
Entry : **SELECT+CROSS+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign FIRE 2
Outline : Records PSX button presses which will be assigned to Atari FIRE2.
Entry : **SELECT+SQUARE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign FIRE 3
Outline : Records PSX button presses which will be assigned to Atari FIRE3.
Entry : **SELECT+TRIANGLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign AUTOFIRE 1
Outline : Records PSX button presses which will be assigned to Atari FIRE1.
Entry : **SELECT+SQUARE+CIRCLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign AUTOFIRE 2
Outline : Records PSX button presses which will be assigned to Atari FIRE2.
Entry : **SELECT+CROSS+CIRCLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign AUTO FIRE 3
Outline : Records PSX button presses which will be assigned to Atari FIRE3.
Entry : **SELECT+TRIANGLE+CIRCLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Assign AUTO LEFT RIGHT
Outline : Records PSX button presses which will be assigned to internal function Atari LEFT RIGHT. When pressed, the left and right directions will toggle repeatedly at the repeat rate.
Entry : **SELECT+SQUARE+TRIANGLE+START, V-A-R-I-A-B-L-E, START**
Variable Expected : Yes

Friendly Name : Toggle Mouse Mode
Outline : Outputs quadrature mouse movements using either Atari ST or Commodore Amiga pin outs.
Entry : **SELECT+UP+RIGHT+DOWN+LEFT+L1+L2+R1+R2**
Variable Expected : No

Friendly Name : Toggle Atari 7800 Fire 3 Mode
Outline : Disables or enables the fire button on pin-5. STs need this to be disabled.
Entry : **SELECT+UP+RIGHT+DOWN+LEFT+SQUARE+CROSS+CIRCLE+TRIANGLE**
Variable Expected : No

Friendly Name : Toggle Left/Right Thumb Stick
Outline : In mouse mode you can use the left or right stick
Entry : **SELECT+UP+RIGHT+DOWN+LEFT+L2+R2**
Variable Expected : No

1.2 COMMANDS - TABLE

COMMAND NAME	LEFT	DOWN	RIGHT	UP	START	-	-	SELECT	SQUARE	CROSS	CIRCLE	TRIANGLE	R1	L1	R2	L2	BYTE 1	BYTE 2
LOAD SETUP 1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1 1	0 0
LOAD SETUP 2	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	2 1	0 0
LOAD SETUP 3	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	4 1	0 0
LOAD SETUP 4	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	8 1	0 0
SAVE SETUP 1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1 1	0 1
SAVE SETUP 2	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	2 1	0 1
SAVE SETUP 3	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	4 1	0 1
SAVE SETUP 4	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	8 1	0 1
ASSIGN FIRE 1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0 9	4 0
ASSIGN FIRE 2	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0 9	8 0
ASSIGN FIRE 3	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0 9	1 0
ASSIGN AUTOFIRE 1	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0 9	6 0
ASSIGN AUTOFIRE 2	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0 9	A 0
ASSIGN AUTOFIRE 3	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0 9	3 0
ASSIGN AUTO LEFT RIGHT	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0 9	9 0
SET AUTO REPEAT RATE	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0 9	F 0
TOGGLE AMIGA/ATARI MOUSE	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1	1	F 1	0 F
TOGGLE FIRE 3 PIN5 OPERATION	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0	F 1	F 0
TOGGLE LEFT/RIGHT THUMB STICK	1	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	F 1	0 3
RESET TO FACTORY	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	F 9	F F

2.0 SETUP MODES - FACTORY DATA

The following data is what is supplied when you receive JeST. You can overwrite it by saving into the relevant slots with your own data. The original data can be restored by entering in the appropriate command, which is deliberately long winded so that you don't accidently do it.

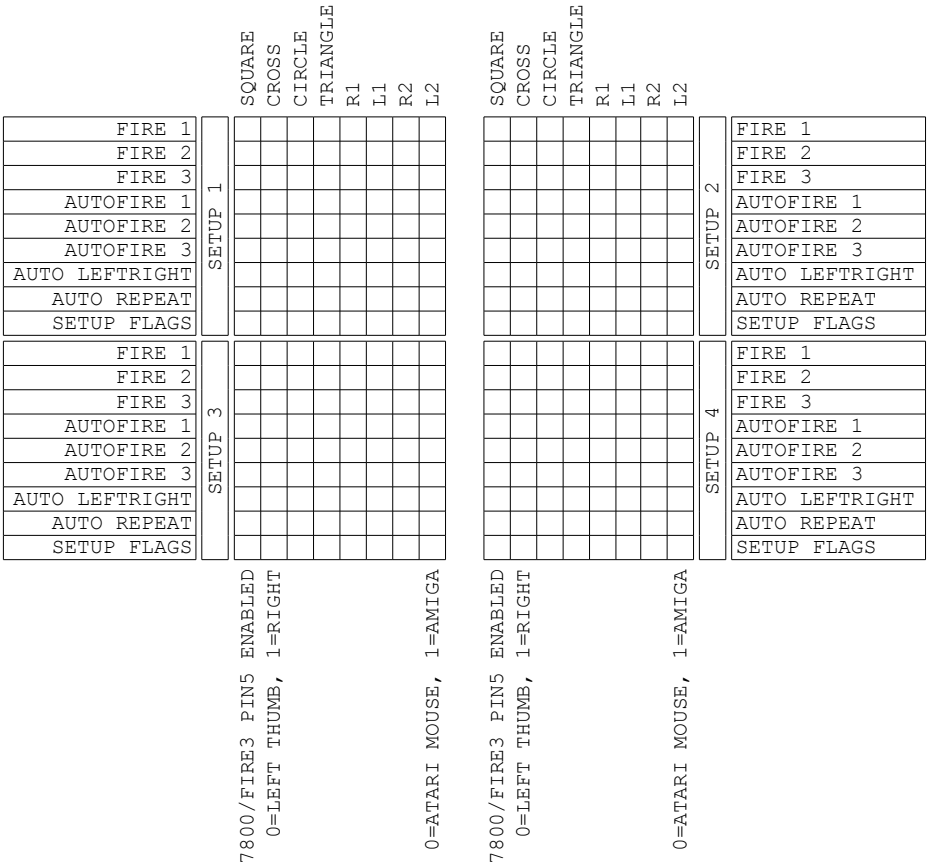
		SQUARE	CROSS	CIRCLE	TRIANGLE	R1	L1	R2	L2									
FIRE 1	SETUP 1	0	1	0	0	0	0	0	0	SETUP 2	0	1	0	0	0	0	0	0	FIRE 1
FIRE 2		1	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	FIRE 2
FIRE 3		0	0	0	1	0	0	0	0		0	0	0	1	0	0	0	0	FIRE 3
AUTOFIRE 1		0	0	0	0	1	1	1	1		0	0	0	0	0	1	0	1	AUTOFIRE 1
AUTOFIRE 2		0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	0	AUTOFIRE 2
AUTOFIRE 3		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	AUTOFIRE 3
AUTO LEFTRIGHT		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	AUTO LEFTRIGHT
AUTO REPEAT		0	0	0	0	0	1	1	1		0	0	0	0	1	1	1	1	AUTO REPEAT
SETUP FLAGS		0	0	0	0	0	0	0	0		0	1	0	0	0	0	0	0	SETUP FLAGS
FIRE 1	SETUP 3	1	1	1	1	0	0	0	0	SETUP 4	0	1	0	0	0	0	0	0	FIRE 1
FIRE 2		0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	FIRE 2
FIRE 3		0	0	0	0	0	0	0	0		0	0	0	1	0	0	0	0	FIRE 3
AUTOFIRE 1		0	0	0	0	1	1	1	1		0	0	0	0	0	1	0	0	AUTOFIRE 1
AUTOFIRE 2		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	1	AUTOFIRE 2
AUTOFIRE 3		0	0	0	0	0	0	0	0		0	0	0	0	1	0	0	0	AUTOFIRE 3
AUTO LEFTRIGHT		0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	AUTO LEFTRIGHT
AUTO REPEAT		0	0	0	0	0	1	1	1		0	0	0	0	0	1	1	1	AUTO REPEAT
SETUP FLAGS		0	0	0	0	0	0	0	0		0	1	0	0	0	0	0	1	SETUP FLAGS

7800/FIRE3 PIN5 ENABLED
0=LEFT THUMB, 1=RIGHT
0=ATARI MOUSE, 1=AMIGA

7800/FIRE3 PIN5 ENABLED
0=LEFT THUMB, 1=RIGHT
0=ATARI MOUSE, 1=AMIGA

2.1 SETUP MODES - USER DATA

This is where you write your own values for reference.



3.0 THE ATARI STYLE JOYSTICK INTERFACE

This interface has been about for many years, and was widely adopted across many platforms. It defines the connector and the pin-outs. Variations on the pin-outs were adopted by other manufacturers, often with the deliberate intention of forcing the user's to buy specific joysticks. Yet the overall interfacing principles remain the same.

There is no communication protocol, and the joysticks themselves are merely switches which pull the appropriate lines to ground. They are totally digital. It is a *digital standard* whereby the only valid states are either on or off.

In the case of the Atari standard, the switches within the joysticks connect the line to GND. They are what is know as *active low*. The lines themselves would commonly be written with an underscore when discussing them in technical documents.

Some interfaces supply +5v (which we need) and others don't. Other interfaces implement two buttons, although button 1 appears universal on pin-6. The pin-outs MUST BE CHECKED before making a connection. The Amstrad and MSX machines are of particular note since the +5v line may appear on another pin, despite the connector being identical.

3.1 9-PIN DTYPE PLUG (FEMALE)

This is the socket (more commonly called *the plug*) which is fitted to the vast majority of Atari style joysticks and mice. It is an industry standard connector and has been in use for perhaps 30-years or more. It is readily available. The *wings* on these connectors though can make them difficult to insert, as in the 1980's the plugs fitted to joysticks were almost always moulded in plastic, taking up the minimum of space.

The wings can be bent back although ideally they should not be cut off, since they are secured together by a flange which meets at the two 3mm holes.

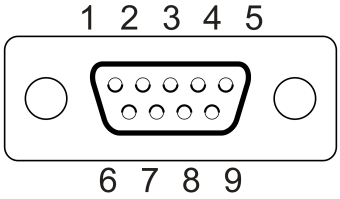


Fig. 9-Pin D-Type Female, Solder Bucket Rear View

1 - UP	**6 - BUTTON 1**
2 - DOWN	**7 - VCC**
3 - LEFT	**8 - GND**
4 - RIGHT	**9 - BUTTON 2**
5 - N/C	

Fig. 9-Pin Atari 2600/ST Style Joy Interface Pin-Outs

3.2 COMMON JOYSTICK PIN-OUTS

There is one often identical similarity between all of the following pin-outs, in that they all utilise a 9-pin D-Type connector. It is identical to the connectors commonly associated with serial interfaces on modern PCs.

The table below details some of the most common pin-outs :-

	2600	7800	Amiga	C64	ST	ZX	MSX	CPC
1	Up	Up	Up	Up	Up	Up	Up	Up
2	Down	Down	Down	Down	Down	Down	Down	Down
3	Left	Left	Left	Left	Left	Left	Left	Left
4	Right	Right	Right	Right	Right	Right	Right	Right
5	-	Button1	-	Y	RESERVED	-	+5v	-
6	Button1	Both B	Button1	Button1	Button1	Button1	Button1	Button1
7	-	-	+5v	-	+5v	-	Button2	Button2
8	GND	GND	GND	GND	GND	GND	Output	GND
9	-	Button2	Button2	X	Button2	-	GND	-
	ATARI	ATARI	ATARI	ATARI	ATARI	ATARI	MSX	AMSTRAD

Fig. Common Joystick Pin Assignments

You will note that most of these pin-outs are almost identical. The most important one is +5v. If you connect an Atari Joystick to an incompatible interface then you risk creating an electrical short. That is not good.

In the majority of cases the fire buttons are swapped around if anything. The Atari 7800 console for example has two fire buttons, yet a standard joystick will often only connect to one of them. The Atari ST, which is designed to utilise a single button, can utilise two.

The table below details the common pin-outs of the same interfaces when connected to a bus mouse. The additional standards to the furthermost right, are included for comparison and completeness :-

	ST	Amiga	Acorn	MS BUS	SERIAL	PS2	USB	
1	XB	YA(V)	XREF	SW2	CD	DATA	+5v	
2	XA	XA(H)	SW1	SW3	RXD	-	DATA-	
3	YA	YB(VQ)	SW2	GND	TXD	GND	DATA+	
4	YB	XB(HQ)	GND	XB	DTR	+5v	GND	
5	-	MB	XDIR	YA	GND	CLK		
6	LB	LB	+5v	YB	DSR	-		
7	+5v	+5v	YREF	SW1	RTS			
8	GND	GND	SW3	+5v	CTS			
9	RB	RB	YDIR	XA	RI			
	BUS	BUS	BUS	BUS	SERIAL	PS2	USB	

Fig. Common Mouse Pin Assignments

The Atari ST and Amiga quadrature bus mouse standards only differ in so far as pin-outs. Otherwise, they are both identical.

4.0 THE PSX CONTROLLER SERIAL BUS

The PSX controller interface is a serial bus of sorts. All communication lines are shared among the connected devices. **ATT** (Attention, Select) is independent of each individual device. Many different devices may share the data lines, yet **ATT** is unique to each device.

It is not necessary to understand any of this information to use JeST. It is merely included to offer a rounded explanation of what's going on.

The signal lines of the interface are as follows (HOST = PSX or *Our Microcontroller* : DEVICE = JoyPad Controller) :-

SIGNAL	DIRECTION			DESCRIPTION
ATT	HOST	▶	DEVICE	Attention, Select, CS, Bus Select
CLK	HOST	▶	DEVICE	Synchronous Clock
CMD	HOST	▶	DEVICE	Command, 8-bit LSB
DAT	HOST	◀	DEVICE	Data, 8-bit LSB
ACK	HOST	◀	DEVICE	Acknowledge, Low for 1 CLK after DAT/CMD

Fig. Outline of PSX Data Signals

ATT, **CLK**, and **ACK** are all pretty self explanatory, and are active low, which is indicated by the underline. **CMD** and **DAT** are counterparts to one another, both transmitting on the same clock pulses. This method of transferring data is know as *asynchronous* and is common with the SPI bus standard.

DAT and **ACK** are the only two where the controller transmits, the rest are receive. The common PSX transmission clock rate is 250Khz (250Kbits/s) or thereabouts. Tolerance is reported to be from about 100Khz up through 500Khz.

The power requirements of a non dual-shock controller (we are NOT using the vibrate feature) are about 30mA at a maximum of 5v. There are reports of the standard being about 3.7v although 5v is considered safe.

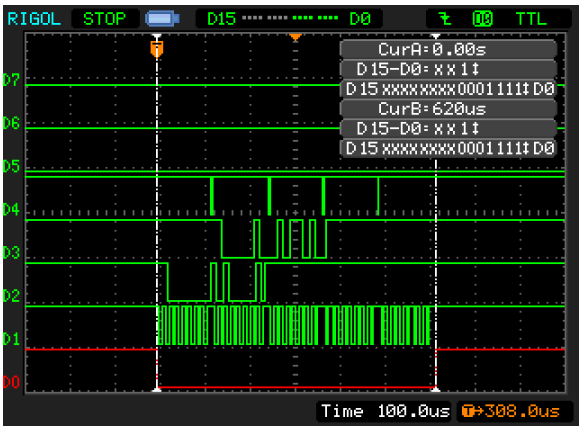


Fig. Example of a 5-Byte Digital Packet

4.1 9-PIN PSX PLUG (FEMALE)

Please refer to the pin-outs table further down the page.

The female connector is fitted to the main PSX unit (host). This is the female connector :-

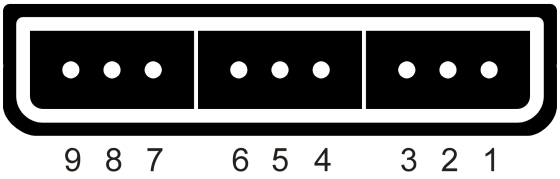


Fig. 9-Pin PSX Socket

4.2 9-PIN PSX PLUG (MALE)

This style of connector is specific to the PSX product range, and is not used on any other product. It is not an industry standard connector. The male version connector is fitted to the controller.

The pin-outs listed here are when looking at the connector face on (ie. NOT the rear solder side). This is the male connector :-

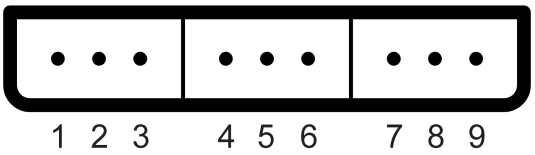


Fig 6. 9-Pin PSX Plug

1 - DATA
2 - COMMAND
3 - N/C
4 - GND
5 - VCC
6 - ATTENTION
7 - CLOCK
8 - N/C
9 - ACKNOWLEDGE

Fig. PSX Style Interface Pin-Outs

5.0 WIRING UP THE JEST PCB

Fig. Male and Female Connectors

			Colour				Colour
1	-	DATA		6	-	ATTENTION	
2	-	COMMAND		7	-	CLOCK	
3	-	N/C		8	-	N/C	
4	-	GND		9	-	ACKNOWLEDGE	
5	-	VCC					

Fig. PSX Style Interface Pin-Outs

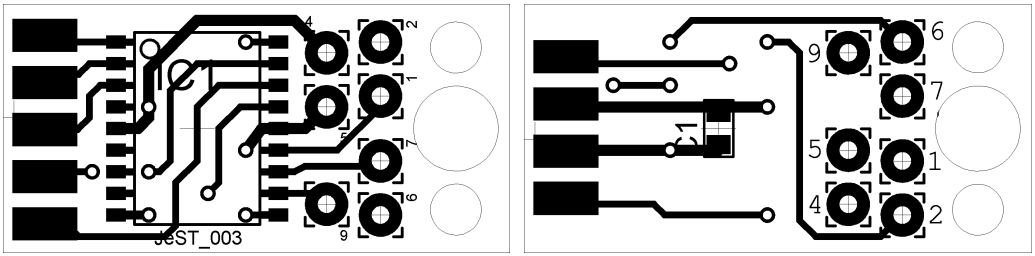


Fig. JeST PCB (Top Side, Bottom Side)

Cable1	Colour	PSX	Signal		MCU	Signal
Brown		1	DAT		13	RB7
Orange		2	CMD		10	RB4
Black		4	GND		5	Vss
Red		5	Vcc		14	Vdd
Yellow		6	ATN		9	RB3
Blue		7	CLK		12	RB6
Green		9	ACK		11	RB5

Fig. MicroController References

6.0 Q&A

Q: When I press the buttons nothing happens.
A: Try disabling pin5/7800/button3. If this is enabled on an ST then it interferes with a reserved pin.

Q: How do I know that power is being supplied to JeST?
A: Press **ANALOGUE** on the controller. The LED will light up.

Q: Is it safe to plug JeST into xyz-computer? How much power does it use?
A: JeST itself draws 5mA. A Sony PS2 'dualshock' controller draws about 20mA. For safety sake, the overall package draws 30mA.

Q: Can I use a wireless PSX controller?
A: There is no reason why not. All JeST sees is a device which responds to it's polling. It is the same situation with PeST (Atari ST PS2 Mouse Interface) whereby if the device supports the standard then it works.

Q: Can I use a steering wheel?
A: No. The wheels have a specific method of communication which is not programmed into JeST.

Q: How do I get this to work with a Sinclair Spectrum?
A: You need to externally supply power to JeST. Ideally you would have the version of JeST which is fitted with a USB cable. Or, you can cut the track leading to pin-7 of the 9-Pin DTYPE, and connect power to holes 4 and 5 on the PCB. Or... You could build a small adapter which sits in-between JeST and your computer, which allows you to inject the power without modifying JeST.

Q: What does FIRE3 do?
A: This is the line for the Atari 7800, and also the Amiga's middle mouse button. It interferes with the Atari-ST which has Pin-5 listed as reserved. Pin-5 on the Atari-ST maintains about 2-3v. I do not know why. FIRE3 is disabled by default. You can enable it, and then save it into one of your setup slots.

Q: Can I disconnect/reconnect the controller?
A: For the most part yes. JeST knows when the controller is disconnected and stops outputting data to the Atari-style joystick port. Occasionally it will crash JeST, in which case you need to disconnect and reconnect the whole interface.

Q: Which setup is loaded at power up?
A: The first setup slot is always loaded at power up. This can be your own setup if you had previously saved it into this slot.

Q: How many times does JeST read the PSX controller per second?
A: It's about 50-times per second. Most of the core timing functions are influenced by this figure, namely the analogue thumbstick and the autofire.

Q: Can the mouse be faster?
A: No. Firstly it becomes uncontrollable when using your thumb, and secondly, JeST is practically working flat out to process the PSX packets. It's an overall balance which was reached while coding.

Q: What does the name JeST actually mean?
A: JeST = **J***oystick* **e***numerator for the Atari* **ST**. PeST = **P***S2* **e***numerator for the Atari* **ST**. WeST = W*inchester* **e***numerator for the Atari* **ST** (this never left development, similar to SatanDisk).

7.0 CREDITS, REFERENCES, AND INFORMATION

This document has been written by techie_alison of the 16-bit Atari scene.

OpenOffice running over Linux Ubuntu 8.04 has been used to write and establish the lay out. It also generated the resulting .PDF file.

CorelDraw 7 (vector based) was used to create the images, running over Windows XP. The resulting images were exported as 1024x1024x16.7 million colour .BMP files. The high colour depth was required to perform the anti-aliasing of the lines.

Linux GNOME GIMP (bitmap based) has been used to convert the resulting .BMP files into .PNG files. The original .BMP files each being some 3Mbytes in size, and the final .PNG files being some 25Kbytes in size.

Credit is given to HeavyStylus (James) who nagged me continually to get this finished. He had one of the first versions which was just a joypad and basic mouse. The version you see here was a complete rewrite to handle the commands and button masks.

Credit is also given to those that show interest in what I do. You help more than you realise.

7.1 WWW ONLINE REFERENCES

The following are websites visited for confirmation of various information :-

PSX Commands

Interfacing a PS2 (PlayStation 2) Controller

Sony Playstation Controller port

Joystick Pinouts