joystick-parport.txt
Linux Joystick parport drivers v2.0
(c) 1998-2000 Vojtech Pavlik <vojtech@ucw.cz>
(c) 1998 Andree Borrmann <a.borrmann@tu-bs.de>
Sponsored by SuSE

---

0. Disclaimer

Any information in this file is provided as-is, without any guarantee that it will be true. So, use it at your own risk. The possible damages that can happen include burning your parallel port, and/or the sticks and joystick and maybe even more. Like when a lightning kills you it is not our problem.

1. Intro

The joystick parport drivers are used for joysticks and gamepads not originally designed for PCs and other computers Linux runs on. Because of that, PCs usually lack the right ports to connect these devices to. Parallel port, because of its ability to change single bits at will, and providing both output and input bits is the most suitable port on the PC for connecting such devices.

2. Devices supported

Many console and 8-bit computer gamepads and joysticks are supported. The following subsections discuss usage of each.

2.1 NES and SNES

The Nintendo Entertainment System and Super Nintendo Entertainment System gamepads are widely available, and easy to get. Also, they are quite easy to connect to a PC, and don't need much processing speed (108 us for NES and 165 us for SNES, compared to about 1000 us for PC gamepads) to communicate with them.

All NES and SNES use the same synchronous serial protocol, clocked from the computer's side (and thus timing insensitive). To allow up to 5 NES and/or SNES gamepads and/or SNES mice connected to the parallel port at once, the output lines of the parallel port are shared, while one of 5 available input lines is assigned to each gamepad.

This protocol is handled by the gamecon.c driver, so that's the one you'll use for NES, SNES gamepads and SNES mice.

The main problem with PC parallel ports is that they don't have +5V power source on any of their pins. So, if you want a reliable source of power for your pads, use either keyboard or joystick port, and make a pass-through cable. You can also pull the power directly from the power supply (the red wire is +5V).

If you want to use the parallel port only, you can take the power is from some data pin. For most gamepad and parport implementations only one pin is needed, and I'd recommend pin 9 for that, the highest data bit. On the other hand, if you are not planning to use anything else than NES / SNES on the port, anything between and including pin 4 and pin 9 will work.

(pin 9) -----> Power

Unfortunately, there are pads that need a lot more of power, and parallel ports that can't give much current through the data pins. If this is your case, you'll need to use diodes (as a prevention of destroying your parallel port), and combine the currents of two or more data bits together.

```
          Diodes
(pin 9) ----|>|-------+------> Power
                      |
(pin 8) ----|>|-------+
                      |
(pin 7) ----|>|-------+
                      |
 <and so on>          :
                      |
(pin 4) ----|>|-------+
```

Ground is quite easy. On PC's parallel port the ground is on any of the pins from pin 18 to pin 25. So use any pin of these you like for the ground.

(pin 18) -----> Ground

NES and SNES pads have two input bits, Clock and Latch, which drive the serial transfer. These are connected to pins 2 and 3 of the parallel port, respectively.

(pin 2) -----> Clock
(pin 3) -----> Latch

And the last thing is the NES / SNES data wire. Only that isn't shared and each pad needs its own data pin. The parallel port pins are:

(pin 10) -----> Pad 1 data
(pin 11) -----> Pad 2 data
(pin 12) -----> Pad 3 data
(pin 13) -----> Pad 4 data
(pin 15) -----> Pad 5 data

Note that pin 14 is not used, since it is not an input pin on the parallel port.

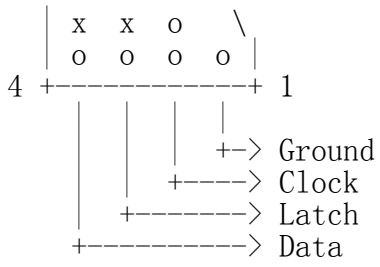
This is everything you need on the PC's side of the connection, now on to the gamepads side. The NES and SNES have different connectors. Also, there are quite a lot of NES clones, and because Nintendo used proprietary connectors for their machines, the cloners couldn't and used standard D-Cannon connectors. Anyway, if you've got a gamepad, and it has buttons A, B, Turbo A, Turbo B, Select and Start, and is connected through 5 wires, then it is either a NES or NES clone and will work with this connection. SNES gamepads also use 5 wires, but have more buttons. They will work as well, of course.

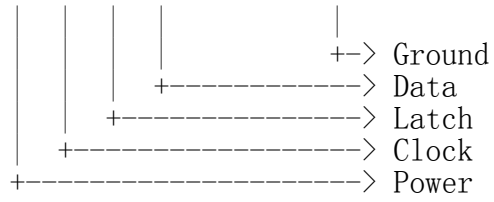
Pinout for NES gamepads

```
          +----> Power
          |
 5 +------+  7
```

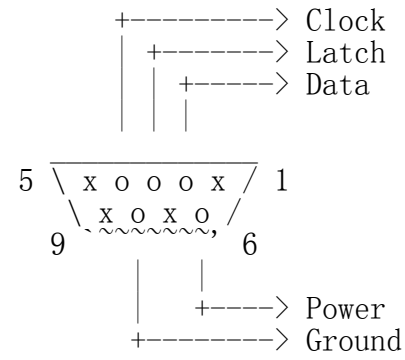
Pinout for SNES gamepads and mice

```
   +----------------------\
 7 | o  o  o  o |  x  x  o  | 1
   +----------------------/
```

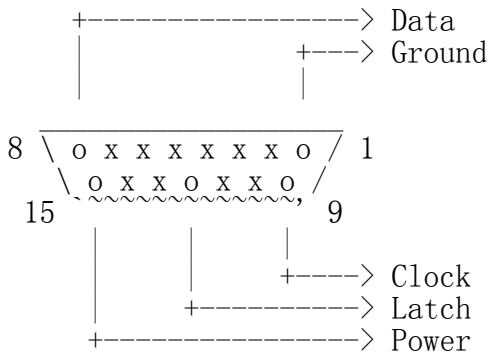




Pinout for NES clone (db9) gamepads



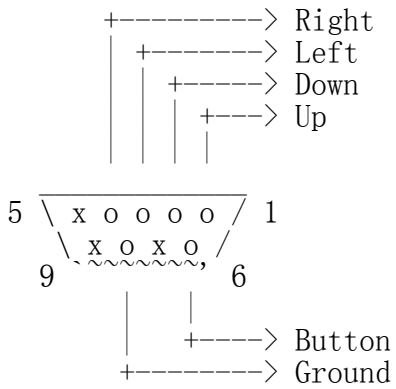
Pinout for NES clone (db15) gamepads



2.2 Multisystem joysticks

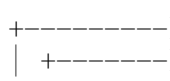
In the era of 8-bit machines, there was something like de-facto standard for joystick ports. They were all digital, and all used D-Cannon 9 pin connectors (db9). Because of that, a single joystick could be used without hassle on Atari (130, 800XE, 800XL, 2600, 7200), Amiga, Commodore C64, Amstrad CPC, Sinclair ZX Spectrum and many other machines. That's why these joysticks are called "Multisystem".

Now their pinout:

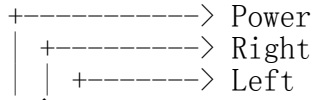


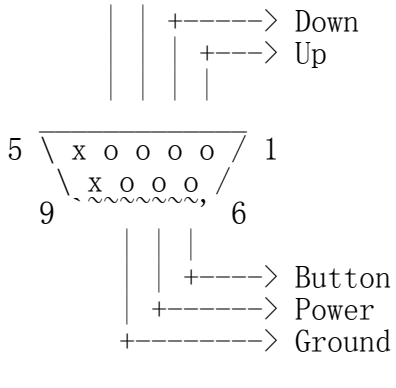
However, as time passed, extensions to this standard developed, and these were not compatible with each other:

Atari 130, 800/XL/XE

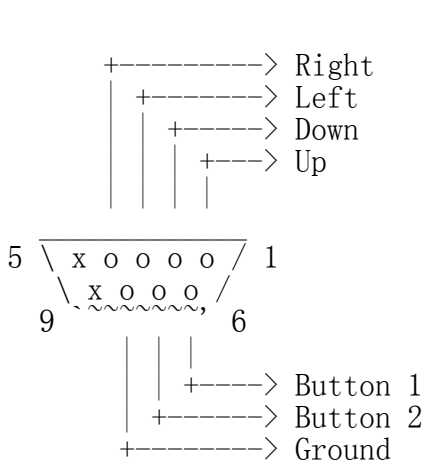


MSX

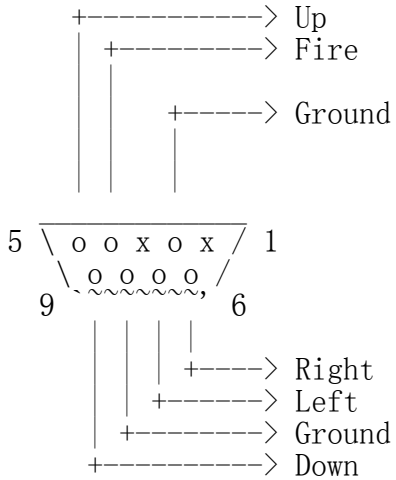


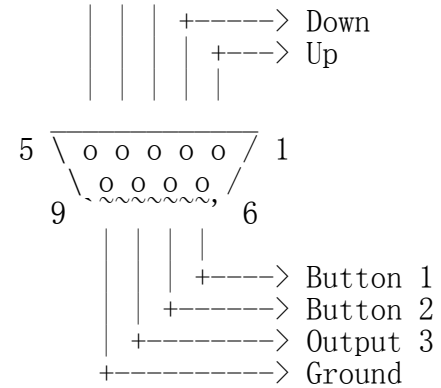


Amstrad CPC

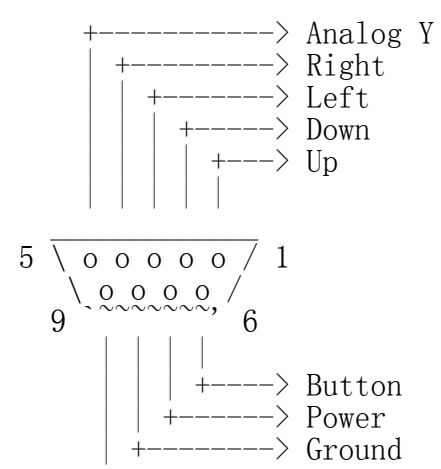


Sinclair Spectrum +2A/+3

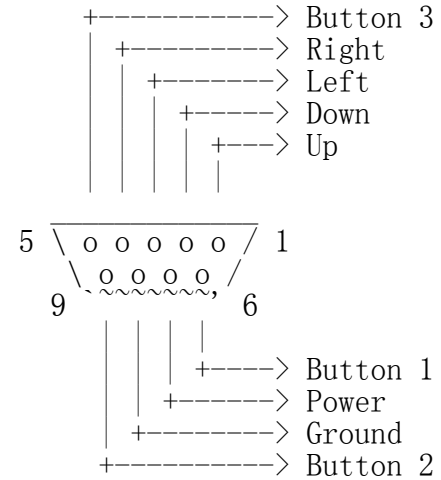




Commodore C64



Amiga 1200



And there were many others.

2.2.1 Multisystem joysticks using db9.c

For the Multisystem joysticks, and their derivatives, the db9.c driver was written. It allows only one joystick / gamepad per parallel port, but the interface is easy to build and works with almost anything.

For the basic 1-button Multisystem joystick you connect its wires to the parallel port like this:

```
(pin  1) -----> Power
(pin 18) -----> Ground

(pin  2) -----> Up
(pin  3) -----> Down
(pin  4) -----> Left
(pin  5) -----> Right
(pin  6) -----> Button 1
```

However, if the joystick is switch based (eg. clicks when you move it), you might or might not, depending on your parallel port, need 10 kOhm pullup resistors on each of the direction and button signals, like this:

```
(pin 2) ------------+------> Up
          Resistor  |
(pin 1) --[10kOhm]--+
```

Try without, and if it doesn't work, add them. For TTL based joysticks / gamepads the pullups are not needed.

For joysticks with two buttons you connect the second button to pin 7 on the parallel port.

```
(pin 7) -----> Button 2
```

And that's it.

On a side note, if you have already built a different adapter for use with the digital joystick driver 0.8.0.2, this is also supported by the db9.c driver, as device type 8. (See section 3.2)

2.2.2 Multisystem joysticks using gamecon.c

For some people just one joystick per parallel port is not enough, and/or want to use them on one parallel port together with NES/SNES/PSX pads. This is possible using the gamecon.c. It supports up to 5 devices of the above types, including 1 and 2 buttons Multisystem joysticks.

However, there is nothing for free. To allow more sticks to be used at once, you need the sticks to be purely switch based (that is non-TTL), and not to need power. Just a plain simple six switches inside. If your joystick can do more (eg. turbofire) you'll need to disable it totally first if you want to use gamecon.c.

Also, the connection is a bit more complex. You'll need a bunch of diodes, and one pullup resistor. First, you connect the Directions and the button the same as for db9, however with the diodes inbetween.

```
            Diodes
(pin 2) -----|<|----> Up
(pin 3) -----|<|----> Down
(pin 4) -----|<|----> Left
```

```
(pin 5) -----|<|----> Right
(pin 6) -----|<|----> Button 1
```

For two button sticks you also connect the other button.

```
(pin 7) -----|<|----> Button 2
```

And finally, you connect the Ground wire of the joystick, like done in this little schematic to Power and Data on the parallel port, as described for the NES / SNES pads in section 2.1 of this file - that is, one data pin for each joystick. The power source is shared.

```
Data    ------------+-----> Ground
          Resistor  |
Power   --[10kOhm]--+
```

And that's all, here we go!

2.2.3 Multisystem joysticks using turbografx.c

The TurboGraFX interface, designed by

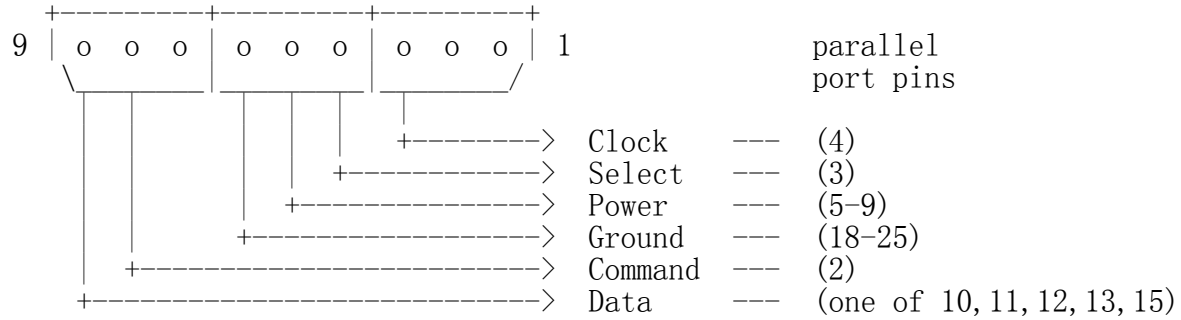
Steffen Schwenke <schwenke@burg-halle.de>

allows up to 7 Multisystem joysticks connected to the parallel port. In Steffen's version, there is support for up to 5 buttons per joystick. However, since this doesn't work reliably on all parallel ports, the turbografx.c driver supports only one button per joystick. For more information on how to build the interface, see

http://www2.burg-halle.de/~schwenke/parport.html

2.3 Sony Playstation

The PSX controller is supported by the gamecon.c. Pinout of the PSX controller (compatible with DirectPadPro):



The driver supports these controllers:

* Standard PSX Pad
* NegCon PSX Pad
* Analog PSX Pad (red mode)
* Analog PSX Pad (green mode)
* PSX Rumble Pad

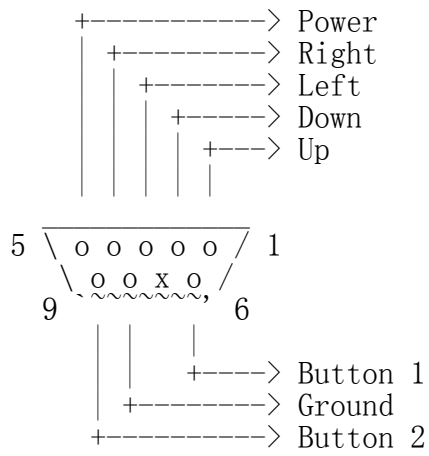
* PSX DDR Pad

2.4 Sega

All the Sega controllers are more or less based on the standard 2-button Multisystem joystick. However, since they don't use switches and use TTL logic, the only driver usable with them is the db9.c driver.

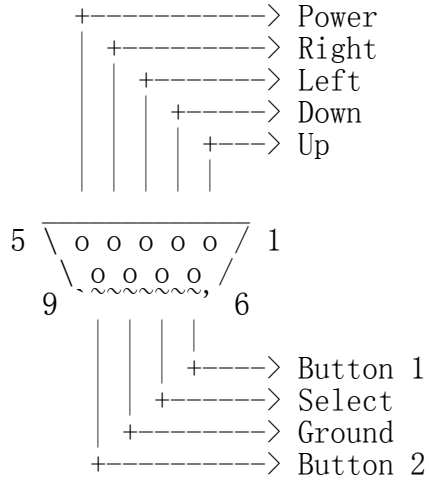
2.4.1 Sega Master System

The SMS gamepads are almost exactly the same as normal 2-button Multisystem joysticks. Set the driver to Multi2 mode, use the corresponding parallel port pins, and the following schematic:



2.4.2 Sega Genesis aka MegaDrive

The Sega Genesis (in Europe sold as Sega MegaDrive) pads are an extension to the Sega Master System pads. They use more buttons (3+1, 5+1, 6+1). Use the following schematic:



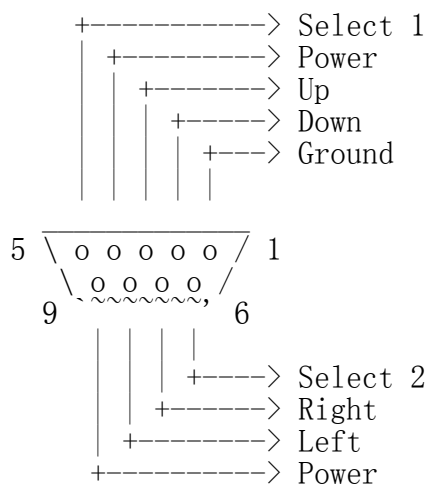
The Select pin goes to pin 14 on the parallel port.

(pin 14) -----> Select

The rest is the same as for Multi2 joysticks using db9.c

2.4.3 Sega Saturn

Sega Saturn has eight buttons, and to transfer that, without hacks like Genesis 6 pads use, it needs one more select pin. Anyway, it is still handled by the db9.c driver. Its pinout is very different from anything else. Use this schematic:



Select 1 is pin 14 on the parallel port, Select 2 is pin 16 on the parallel port.

(pin 14) -----> Select 1
(pin 16) -----> Select 2

The other pins (Up, Down, Right, Left, Power, Ground) are the same as for Multi joysticks using db9.c

3. The drivers

There are three drivers for the parallel port interfaces. Each, as described above, allows to connect a different group of joysticks and pads. Here are described their command lines:

3.1 gamecon.c

Using gamecon.c you can connect up to five devices to one parallel port. It uses the following kernel/module command line:

gamecon.map=port,pad1,pad2,pad3,pad4,pad5

Where 'port' the number of the parport interface (eg. 0 for parport0).

And 'pad1' to 'pad5' are pad types connected to different data input pins (10,11,12,13,15), as described in section 2.1 of this file.

The types are:

Type	Joystick/Pad
0	None

1	SNES pad
2	NES pad
4	Multisystem 1-button joystick
5	Multisystem 2-button joystick
6	N64 pad
7	Sony PSX controller
8	Sony PSX DDR controller
9	SNES mouse

The exact type of the PSX controller type is autoprobed when used, so hot swapping should work (but is not recommended).

Should you want to use more than one of parallel ports at once, you can use gamecon.map2 and gamecon.map3 as additional command line parameters for two more parallel ports.

There are two options specific to PSX driver portion. gamecon.psx_delay sets the command delay when talking to the controllers. The default of 25 should work but you can try lowering it for better performance. If your pads don't respond try raising it until they work. Setting the type to 8 allows the driver to be used with Dance Dance Revolution or similar games. Arrow keys are registered as key presses instead of X and Y axes.

3.2 db9.c

Apart from making an interface, there is nothing difficult on using the db9.c driver. It uses the following kernel/module command line:

```
db9.dev=port,type
```

Where 'port' is the number of the parport interface (eg. 0 for parport0).

Caveat here: This driver only works on bidirectional parallel ports. If your parallel port is recent enough, you should have no trouble with this. Old parallel ports may not have this feature.

'Type' is the type of joystick or pad attached:

Type	Joystick/Pad
0	None
1	Multisystem 1-button joystick
2	Multisystem 2-button joystick
3	Genesis pad (3+1 buttons)
5	Genesis pad (5+1 buttons)
6	Genesis pad (6+2 buttons)
7	Saturn pad (8 buttons)
8	Multisystem 1-button joystick (v0.8.0.2 pin-out)
9	Two Multisystem 1-button joysticks (v0.8.0.2 pin-out)
10	Amiga CD32 pad

Should you want to use more than one of these joysticks/pads at once, you can use db9.dev2 and db9.dev3 as additional command line parameters for two more joysticks/pads.

3.3 turbografx.c

The turbografx.c driver uses a very simple kernel/module command line:

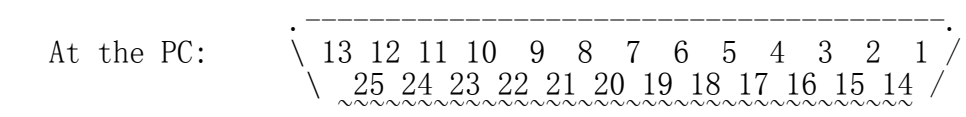
turbografx.map=port,js1,js2,js3,js4,js5,js6,js7

Where 'port' is the number of the parport interface (eg. 0 for parport0).

'jsX' is the number of buttons the Multisystem joysticks connected to the interface ports 1-7 have. For a standard multisystem joystick, this is 1.

Should you want to use more than one of these interfaces at once, you can use turbografx.map2 and turbografx.map3 as additional command line parameters for two more interfaces.

3.4 PC parallel port pinout



Pin	Name	Description
1	/STROBE	Strobe
2-9	D0-D7	Data Bit 0-7
10	/ACK	Acknowledge
11	BUSY	Busy
12	PE	Paper End
13	SELIN	Select In
14	/AUTOFD	Autofeed
15	/ERROR	Error
16	/INIT	Initialize
17	/SEL	Select
18-25	GND	Signal Ground

3.5 End

That's all, folks! Have fun!