SONY PLAYSTATION® CONTROLLER INFORMATION

What you will find here

- The Playstation Controller Pinouts
- The PSX Controller Signals
- The PSX Controller Data
- A Circuit to Emulate a PSX Controller in 74XX Logic
- A Microcontroller to Emulate a PSX Controller

The Playstation Controller Pinouts

```
       LOOKING AT THE PLUG
       -------------------------------
PIN 1->| o  o  o | o  o  o | o  o  o |
       \_____________________________/
```

PIN # USAGE

1. DATA
2. COMMAND
3. N/C (9 Volts unused)
4. GND
5. VCC
6. ATT
7. CLOCK
8. N/C
9. ACK

DATA
: Signal from Controller to PSX.
This signal is an 8 bit serial transmission synchronous to the falling edge of clock (That is both the incoming and outgoing signals change on a high to low transition of clock. All the reading of signals is done on the leading edge to allow settling time.)

COMMAND
: Signal from PSX to Controller.
This signal is the counter part of DATA. It is again an 8 bit serial transmission on the falling edge of clock.

VCC
: VCC can vary from 5V down to 3V and the official SONY Controllers will still operate. The controllers outlined here really want 5V.
The main board in the PSX also has a surface mount 750mA fuse that will blow if you try to draw to much current through the plug (750mA is for both left, right and memory cards).

ATT

ATT is used to get the attention of the controller.
This signal will go low for the duration of a transmission. I have also seen this pin called Select, DTR and Command.

CLOCK

Signal from PSX to Controller.
Used to keep units in sync.

ACK

Acknowledge signal from Controller to PSX.
This signal should go low for at least one clock period after each 8 bits are sent and ATT is still held low. If the ACK signal does not go low within about 60 us the PSX will then start interogating other devices.

It should also be noted that this is a bus of sorts. This means that the wires are all tied together (except select which is seperate for each device). For the CLK, ATT, and CMD pins this does not matter as the PSX is always the originator. The DATA and ACK pins however can be driven from any one of four devices. To avoid contentions on these lines they are open collectors and can only be driven low.

The PSX Controller Signals

All transmissions are eight bit serial LSB first. All timing in the PSX controller bus is syncronous to the falling edge of the clock. One byte of the transmissions will look kinda like this.

```
           |BIT 0|BIT 1|BIT 2|BIT 3|BIT 4|BIT 5|BIT 6|BIT 7|
CLOCK -----___---___---___---___---___---___---___---___-----------

DATA  -----000000111111222222333333444444555555666666777777--------
              *     *     *     *     *     *     *     *
CMND  -----000000111111222222333333444444555555666666777777--------

ACK   ----------------------------------------------------------__-
```

The logic level on the data lines is changed by the transmitting device on the falling edge of clock. This is then read by the receiving device on the leading edge (at the points marked *) allowing time for the signal to settle. After each COMMAND is recieved by a selected controller, that controller needs to pull ACK low for at least one clock tick. If a selected controller does not ACK the PSX will assume that there is no controller present.

When the PSX wants to read information from a controller it pulls that devices ATT line low and issues a start command (0x01). The Controller Will then reply with its ID (0x41=Digital, 0x23=NegCon, 0x73=Analogue Red LED, 0x53=Analogue Green LED). At the same time as the controller is sending this ID byte the PSX is transmitting 0x42 to request the data. Following this the COMMAND line goes idle and the controller transmits 0x5A to say "here comes the data".

This would look like this for a digital controller

```
ATT    -______________________________________________________________
          |   Byte 1     |    |   Byte 2     |    |   Byte 3     |
CLOCK  ---_-_-_-_-_-_-_-_-----_-_-_-_-_-_-_-_-----_-_-_-_-_-_-_-_-----
               0xFF                0x41                0x5A
DATA   -------------------------__________--__----__--__----__--__----
               0x01                0x42
```

```
CMND   -----_____________-----__--________--__------------------------

ACK    --------------------__-------------------__-----------------__-
```

After this command initiation proccess the controller then sends all its data bytes (in the case of a digital controller there is only two). After the last byte is sent ATT will go high and the controller does not need to ACK.

The data transmision for a digital controller would look like this (where A0,A1,A2...B6,B7 are the data bits in the two bytes).

```
ATT    _______________________________________-------
          |   Byte 4      |    |   Byte 5      |
CLOCK  ---_-_-_-_-_-_-_-_-----_-_-_-_-_-_-_-_--------

DATA   ---D0D1D2D3D4D5D6D7----E0E1E2E3E4E5E6E7-------

CMND   ----------------------------------------------
                                         ***
ACK    --------------------__------------------------

NOTE: No ACK.
```

The PSX Controller Data

Below are five tables that show the actual bytes sent by the controllers

```
Standard Digital Pad

BYTE    CMND    DATA

 01     0x01    idle
 02     0x42    0x41
 03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
 04     idle    data    SLCT           STRT UP   RGHT DOWN LEFT
 05     idle    data    L2   R2    L1  R1   /\   O    X    |_|

All Buttons active low.


NegCon

BYTE    CMND    DATA

 01     0x01    idle
 02     0x42    0x23
 03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
 04     idle    data                   STRT UP   RGHT DOWN LEFT
 05     idle    data                   R1   A    B
 06     idle    data    Steering  0x00 = Right 0xFF = Left
 07     idle    data    I  Button 0x00 = Out   0xFF = In
 08     idle    data    II Button 0x00 = Out   0xFF = In
 09     idle    data    L1 Button 0x00 = Out   0xFF = In
```

All Buttons active low.

Analogue Controller in Red Mode

```
BYTE    CMND    DATA

 01     0x01    idle
 02     0x42    0x73
 03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
 04     idle    data    SLCT JOYR JOYL STRT UP   RGHT DOWN LEFT
 05     idle    data    L2   R2   L1   R1   /\   O    X    |_|
 06     idle    data    Right Joy 0x00 = Left  0xFF = Right
 07     idle    data    Right Joy 0x00 = Up    0xFF = Down
 08     idle    data    Left Joy  0x00 = Left  0xFF = Right
 09     idle    data    Left Joy  0x00 = Up    0xFF = Down
```

All Buttons active low.

Analogue Controller in Green Mode

```
BYTE    CMND    DATA

 01     0x01    idle
 02     0x42    0x53
 03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
 04     idle    data                   STRT UP   RGHT DOWN LEFT
 05     idle    data    L2   L1   |_|  /\   R1   O    X    R2
 06     idle    data    Right Joy 0x00 = Left  0xFF = Right
 07     idle    data    Right Joy 0x00 = Up    0xFF = Down
 08     idle    data    Left Joy  0x00 = Left  0xFF = Right
 09     idle    data    Left Joy  0x00 = Up    0xFF = Down
```

All Buttons active low.

PSX Mouse
(credit to T.Fujita
 http://www.keisei.tsukuba.ac.jp/~kashima/games/ps-e.txt)

```
BYTE    CMND    DATA

 01     0x01    idle
 02     0x42    0x12
 03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
 04     idle    0xFF
 05     idle    data         L    R
 06     idle    data    Delta Vertical
 07     idle    data    Delta Horizontal
```

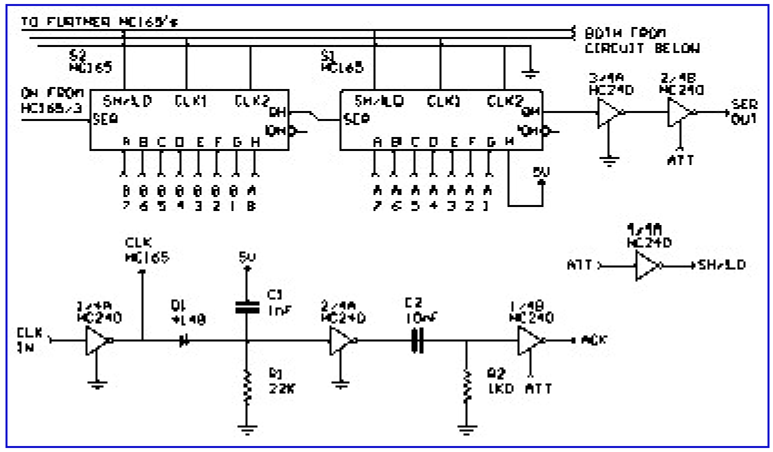
All Buttons active low.

A Circuit to Emulate a PSX Controller in 74XX Logic

This circuit can be set up to emulate a digital controller, an analogue controller (in either mode) or a NegCon. The circuit uses six 74XX IC's to emulate a digital controller. To emulate an analogue controller a further four 74XX IC's and four A/D converters are needed. To emulate a NegCon you also need four A/D's and four extra 74XX over the digital controllers six chips.

How the circuit works

When ATT is pulled LOW by the PSX, inverter 4/4A will pull SH/!LD on the 74HC165's HIGH. This will load the data at their serial input pins (Only two of five IC's are shown). Inverter 1/4A supplies the clocking for the HC165's so that on each falling edge of the PSX's supplied CLK the next bit of data in the HC165's is shifted out. This data out is gated by the ATT signal on inverter 2/4B, this is so only the selected device can drive the bus.



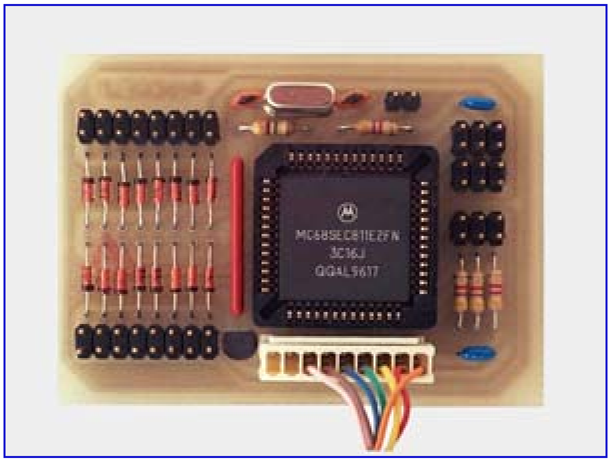
The data loaded into the HC165's is directly related to the data shown in the timming diagrams in the section above. Looking back at the diagram shows the first byte as 0xFF. This means that A1 through to A8 would all be tied HIGH. The next byte is the controller ID. For the digital controller this is 0x41 so B1-8 are HLLLLLHL respectivly. The next byte is the DATA READY command 0x5A so C1-8 are HLHLLHLH. The final two bytes are the button presses and should be set HIGH (through a pull up) when the button is not pressed or LOW if the button is pressed. (NB E8 is SER IN on the last HC165)

The Ack signal is provided by the missing pulse detector built around inverters 1/4A, 2/4A and 1/4B. Diode D1 only allows inverter 1/4A to charge up C1 when CLK is low. When CLK goes HIGH for longer than the time set by R1/C1, inverter 2/4A will go HIGH. This HIGH going signal is coupled to inverter 1/4B via C2 which causes its output to pulse LOW. This signal is also gated by the ATT signal (again to avoid bus contention).

A Microcontroller to Emulate a PSX Controller

The micro controller version uses the Motorola 68HC11 (no one presented a compelling argument to use anything different). The code is currently set up to assemble for an E2 but can be changed to run on any 52 pin version with EEPROM (I made sure that I didn't use interrupts or use more than 256 bytes).

Clicking on either the picture to the left or the overlay below will take you to a larger version of the picture. In the large version clicking on any component will take you to that part in the parts list.



How the circuit works

The circuit uses a Motorola MC68HC11 to do everything. It reads the state of the sixteen N/O (normaly open) switch inputs on the left. It reads the values of the four analog inputs on the right. It then sends this data out one of its two seriel ports.

The four jumpers in the circuit affect how the circuit works. J1 at the top of the board is used to put the HC11 into one of two opperating modes. With J1 open the microcontroller is in single chip mode with it closed the micro is in special bootstrap mode.

J2,3 and 4 select what type of PSX controller to emulate as shown in the table below.

MODE	J2	J3	J4
Digital	O	O	O
Analog RED	O	O	C
NegCon	O	C	O
Analog GREEN	O	C	C
Future use	C	X	X
O = Open, C = Closed, X = Dont Care			

The four analog inputs in the top right hand corner of the circuit are designed for hooking up potentiometers. The size of the pots is not critical but a 10K to 50K pot would be sensible. The three wires for each pot are WIPER, VCC and GND as shown below.

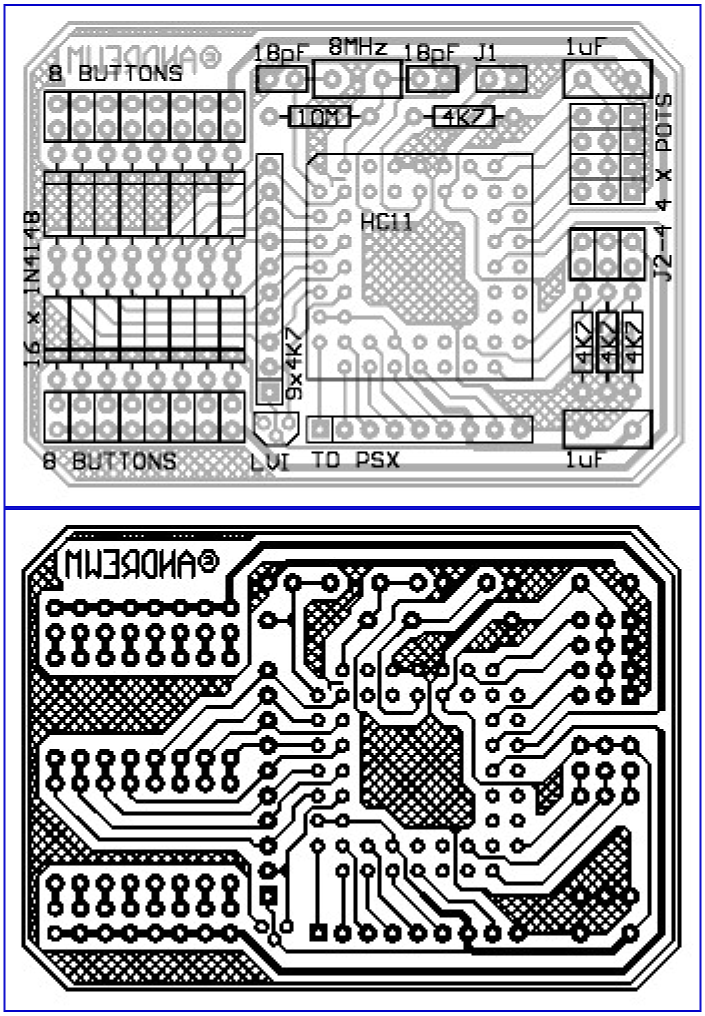
```
  ____________________
  |                 ||
  |                 \/
  |           /\  /\  /\  /\
  |          /  \/  \/  \/  \
  |          |               |
  |          |               |
  O          O               O
PIN 3      PIN 2           PIN 1
```

The nine pin connector at the bottom of the circuit goes to the playstation as well as being used to program the microcontroller. Its pin assignments are shown below.

Pin #	Usage
1	SCI RX FOR RS-232 comms (not used by psx)
2	SCI TX "
3	DATA (pin 1 on PSX)
4	CLOCK (pin 2 on PSX)
5	COMMAND (pin 7 on PSX)
6	ATT (pin 6 on PSX)
7	VCC (pin 5 on PSX)
8	ACK (pin 9 on PSX)
9	GND (pin 4 on PSX)

Finaly the LVI (low voltage inhibitor) at the bottom of the circuit is used to keep the HC11 in a state of RESET when there is not enough voltage to run it safely. It is fairly safe to leave this out of the circuit if you can't get hold of one but be warned you may pop that fuse if you don't include it.

The PCB



The overlay above shows the PCB and all the components on it. The parts and their functions are described below.

The parts

U1	MC68HC11E2
U2	MC 34064 Low voltage inhibitor
XTAL	8Mhz crystal
C1, C2	18pF ceramic cap
C3, C4	1uF mono or MKT cap
R1	10M res
R2- R5	4K7 res
RP1	4K7 x 9 sip res pack
CON1	9 way sil connector
D1- D16	1N4148 diode
J1	Boot mode jumper

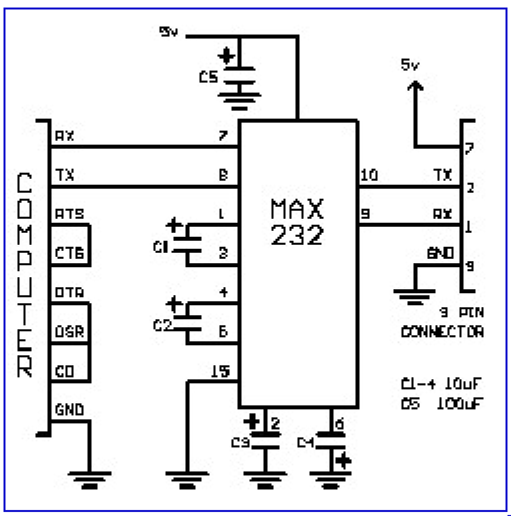
J2- J4	Controller mode jumpers
POT1- 4	Analoge inputs
Button 1- 16	Header for 16 N/O momentary switches

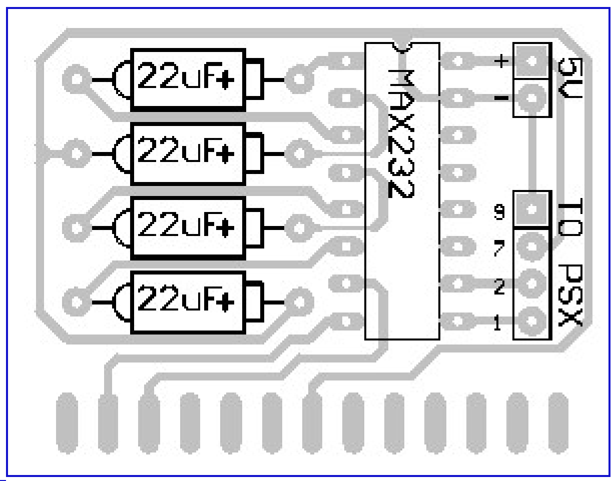
The software

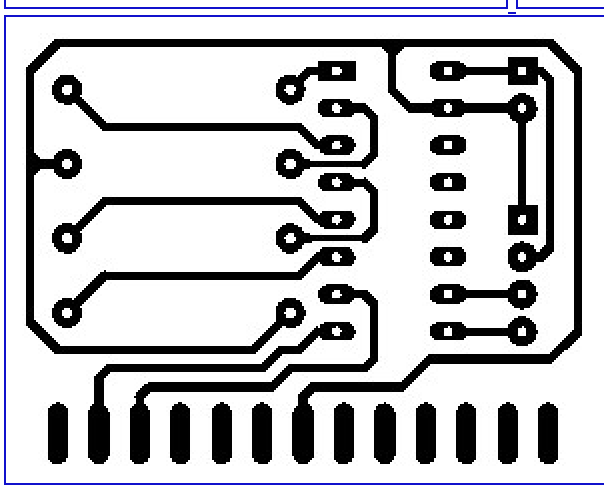
The software is for the microcontroller is VERY heavily commented and should not need any explination. It can be downloaded with the PCB (in auto/easytrax format) here PSXCONT.ZIP

Downloading the software to the HC11

Also included in the ZIP file above is a program called EELOADER.EXE. This is an IBM executable that can be used to download the code into the HC11. The first two pins on the nine pin connector are a 5V RS232 port. This can be connected to any seriel port on your IBM compatable PC through a MAX232 as shown below.







To use it connect it all to a seriel port on your PC and type `EELOAD PSXCONT.S19 /Cx` where x is the com port the controller is on. Then follow the instructions on the screen.

Please read the disclaimer

Maintained by andrewm@quicknet.com.au
Last Updated August 13, 1998.
This pages URL is HTTP://home.quicknet.com.au/andrewm/electron/psxcont.html