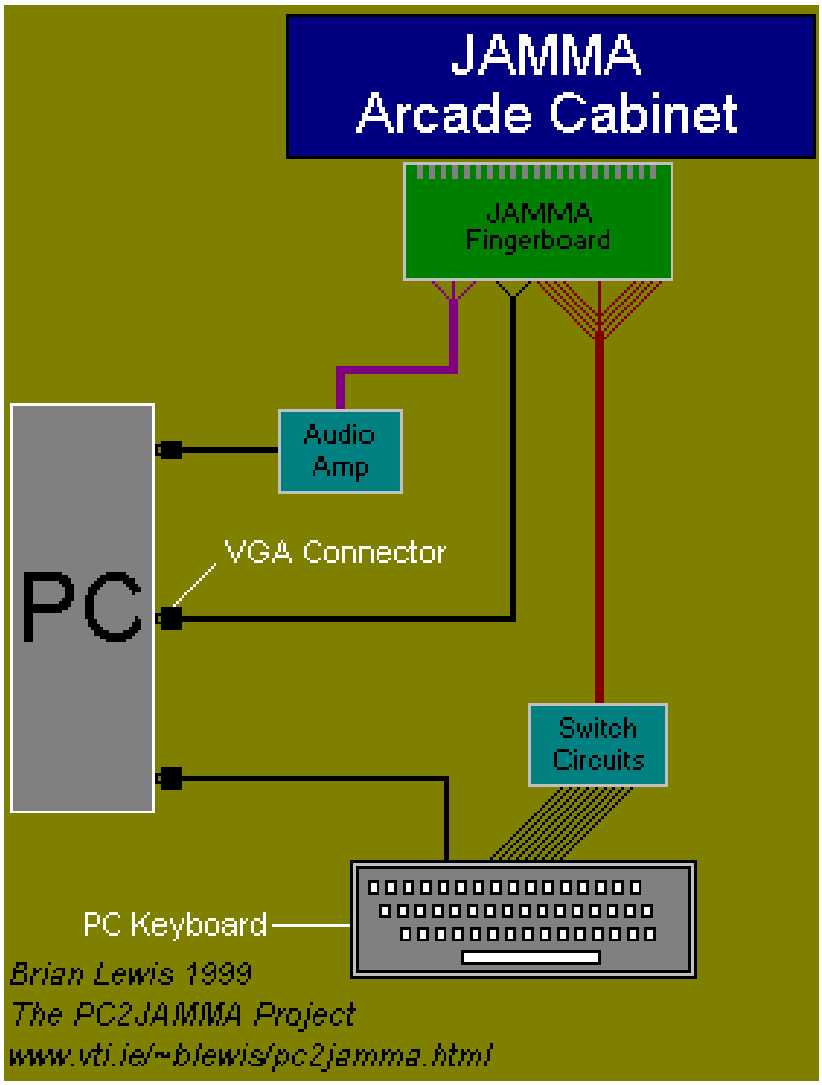
PC2JAMMA project
Before reading through/undertaking this project, I strongly suggest you read the PC2JAMMA FAQ first

This project mutated out of something else,
originally I had planned to take my existing arcade game boards and have them selectable from the front of a cabinet, just to eliminate board swapping.
Turned out to be a fair amount of work, and not 100% guarenteed to work with all my boards.
So I started looking at using a PC emulator instead. It then turned into the MAME2JAMMA project and finally became the **PC2JAMMA Project**.
The project differs from most PC Arcade setups as it alters the PC to fit JAMMA specs, rather than altering the cabinet to fit PC specs. It requires a small amount of hardware, and a special frontend which is capable of running on an arcade monitor.

To give you some idea what the project is about, it's basically a harness; with a PC on one end and a JAMMA arcade cabinet on the other
Something a bit like this.



If you're undertaking this project...
Some experience with electronics is preferable, though you can get by with very little.
See the 'Electronics Free Version' if you don't fancy making up the circuits required.

For this project I only bought the bare essentials of a PC, i.e. mother board, P266, 64MB, hard disk, sound and video card.
This keeps the cost of the project nice and low.

What you need

There are 4 different versions of the PC2JAMMA harness outlined here
The differences between them are really only to do with the number of controls and how they're wired in.

The 4 different versions are:-
1 Player 3 button using an adapted PC Joypad
1 Player 3 button using just the keyboard
2 Player 3 button (per player) using just the keyboard
and the 'Electronics Free' Version. For each of the 4 versions you'll need:-

---

1 JAMMA Arcade Cabinet
1 PC
1 PC Keyboard
1 JAMMA fingerboard (optional for 'Electronics Free Version)
1 VGA connector (plug) or VGA cable
1 x Mini Stereo Jack (for soundcard output)
and some wire

The additional components needed for the different versions are outlined in the **Controls** section.

Making the Adapter

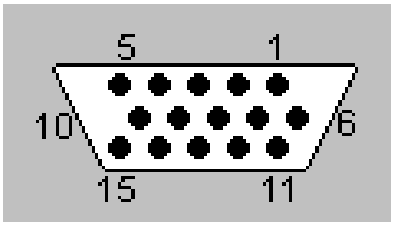
I'll break construction of the adapter down into 3 parts:-
Video, Audio and Controls.

Video

Simple enough, requires no hardware whatsoever.
It's easy to get MAME running on an arcade monitor - simply run MAME with the ***-monitor arcade*** option.

The front end (ArcadeOS.EXE) also allows other emulators without a **-NTSC** option to be connected directly to an arcade monitor.

It does this by reprogramming the VGA timing registers to match a standard res arcade monitor.
Check out the video cards page for more info All you need to do is wire the correct output from the VGA port to the correct pin on the JAMMA fingerboard
The standard 15pin VGA pinout is as follows
(looking at the connector on your VGA card)



1 - Red
2 - Green
3 - Blue
4 - Monitor ID *
5 - Ground
6 - Red Ground
7 - Green Ground
8 - Blue Ground
9 - Keyway (No pin)
10 - Sync Ground
11 - Monitor ID *
12 - Monitor ID *
13 - Horizontal Sync
14 - Verical Sync
15 - Monitor ID *

You can ignore all the pins marked *, as we won't be using them. Don't forget to wire in all the grounds, wire the first one to Video Ground on the JAMMA fingerboard, then solder a short wire from this across to one of the Ground pins on the fingerboard. This just makes sure all your grounds are common.

As for getting composite sync from separate horizontal and vertical sync (which is what VGA puts out), you should run them through and XOR gate (7486) but I cheat - and just twist the wires together.

Audio

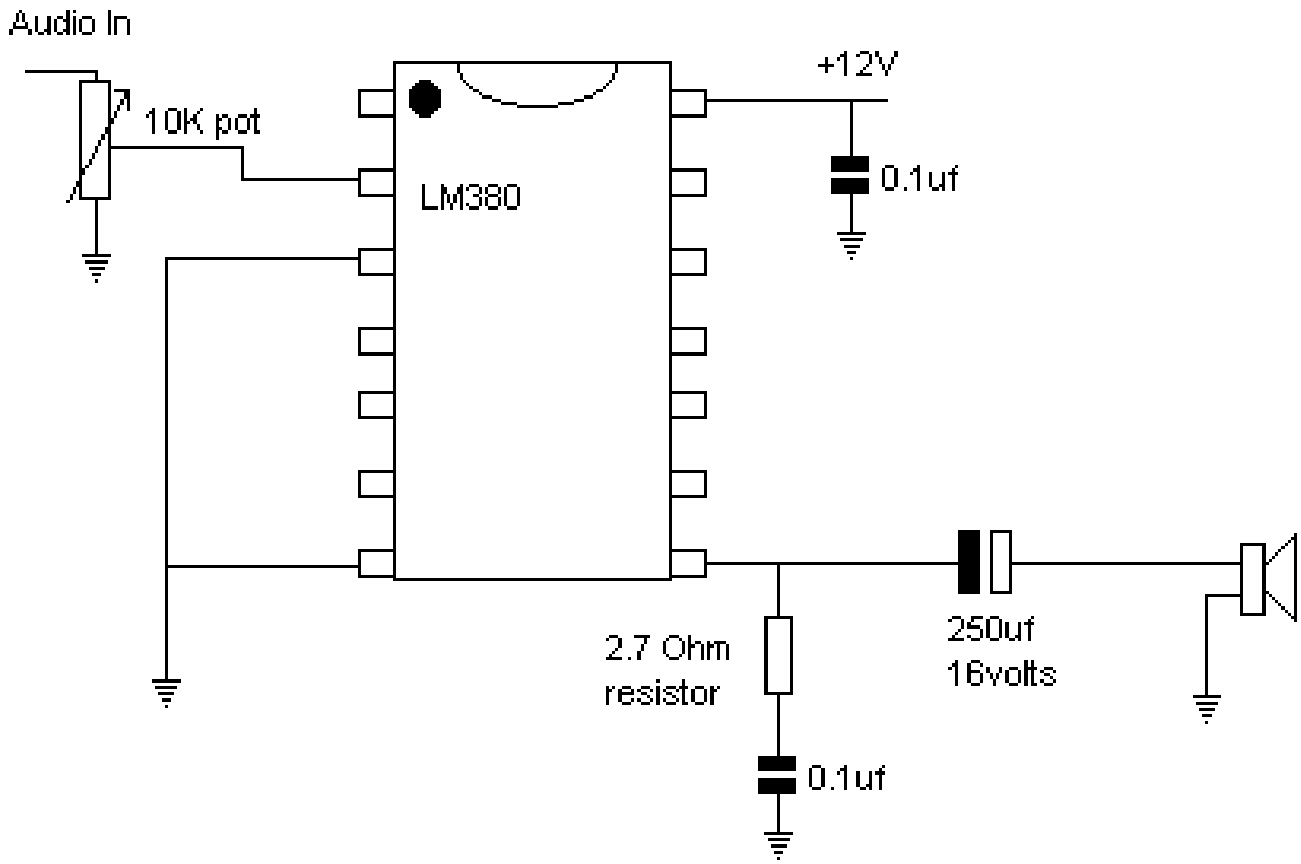
The only thing we need to do here is amplify the output of the soundcard.
You can either build the audio amp described below, buy a kit or ready build amp,
or replace the speakers in your arcade cabinet with powered PC speakers.

The audio amp described here is a simple circuit based around the LM380,
and is very easy to build
Simply take the output from the soundcard, feed it into the amp and connect the speaker output to the JAMMA fingerboard.
('Speaker +' to the output of the audio amp, 'Speaker -' to ground)

If possible, power the audio AMP from the PC's powersupply. If this is not an option, simply power it from the arcade cabinet.

(Click on image for more detailed diagram)

Audio Amp::Circuit Diagram



You can see a photo of what it looks like made up on a bit of stripboard here
If you'd like more details on how to set the circuit out on a bit of stripboard, click here

NOTE: If you use the arcade's power supply, and you have a soundcard with an FM chip
- make sure you set

beepfm=0

in ArcadeOS.CFG.
See ArcadeOS.DOC (in ArcadeOS download) for details.

Controls

This is the most complicated part of the project, allthough the electronics are still fairly straightforward If the idea of making circuits doesn't appeal to you, check out the 'Electronics Free Version' of the project.

The Keyboard.

Each of the 4 versions of the harness involve soldering onto the keyboard main IC
so we'll start by taking the keyboard to bits....
Take the outer casing off your keyboard , you'll see that it's basically a very small circuit board with 2 thin 'ribbon' like connectors going into it from the main body of the keyboard.
Take the main part of the keyboard to bits, you'll see it consists of pieces clear plastic with conductive 'tracks' on their surfaces.
This forms a matrix, which uniquely identifies a key when 2 lines are connected together. These lines go back to the small circuit board, one will have a lot of pins/lines, the other less.

All we have to do is trick the keyboard into thinking that a key has been pressed when a button is pressed inside the cabinet - which means we have to connect the 2 lines which identify the correct key when a joystick is moved or a button pressed in the arcade cabinet.

Controls in JAMMA (and most other) cabinets work like this:-

Each control line at the JAMMA edge connector is 'open' (not connected to anything) until a button is pressed/coin inserted or a joystick moved.
When this happens this line is connected to ground.

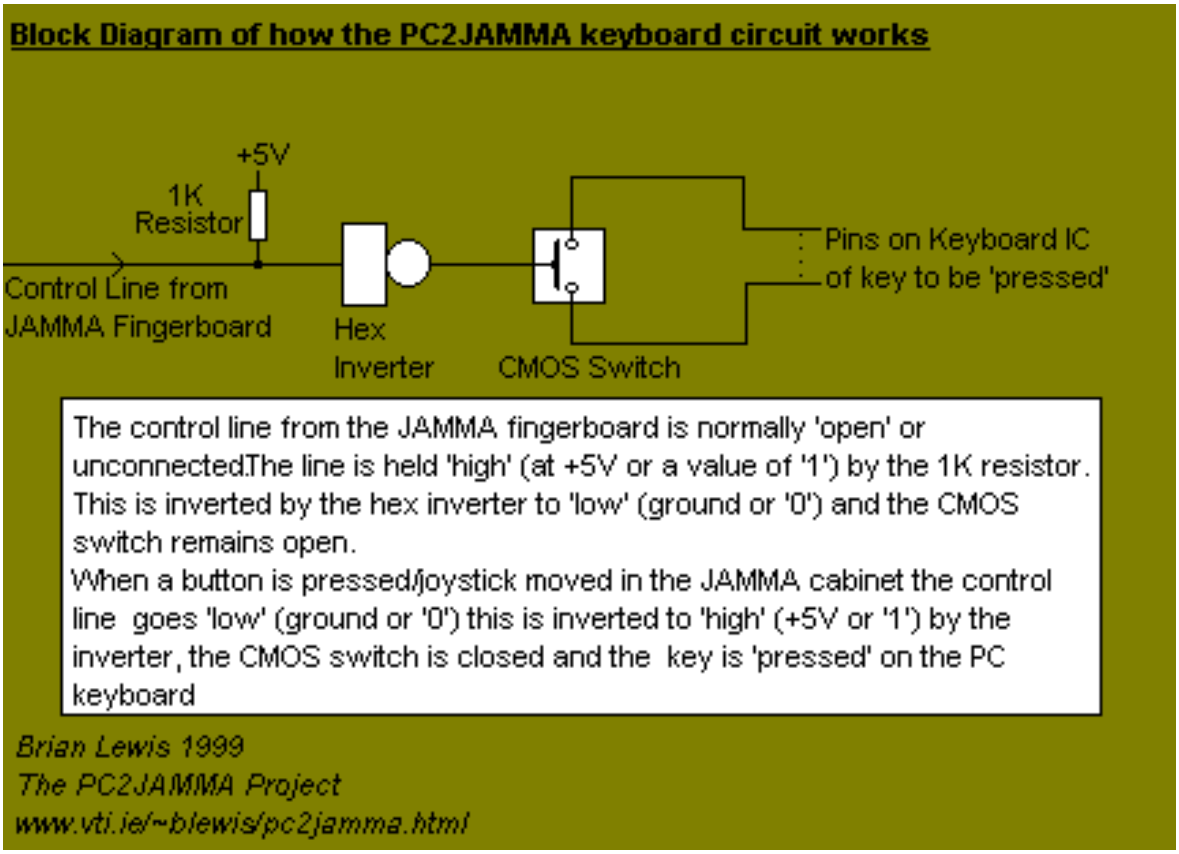
We need to convert this open/ground state into a key press.

Start by identifying the key you need pressed (say 1 Player start, key 1). Trace the 2 lines back from this key to the keyboard's circuit board. Solder on a wire to each of the 2 pins you've identified by tracing the line back.
If you touch these wires together - you should get a keypress.

IMPORTANT: Do NOT solder onto the plastic, you'll just ruin your keyboard
trace each line all the way back to the IC and solder onto the pins of that chip

Now we need to make a circuit which will connect these 2 wires when a button is pressed, coin inserted or a joystick moved in the arcade cabinet.
To do this we take a control line we're interested in from the JAMMA fingerboard into a small circuit with a resistor holding the line high (+5V) then through an inverter (7404) and finally to a switch (4066). The 2 wires from the keyboard will be connected to and acted upon by the switch.

Here's a block diagram detailing how the circuit works for one control line
(Click on it for more legible version)



You can see what a circuit for 3 lines (Player1 Start, Player2 Start and Coin1) looks like made up on a bit of stripboard here

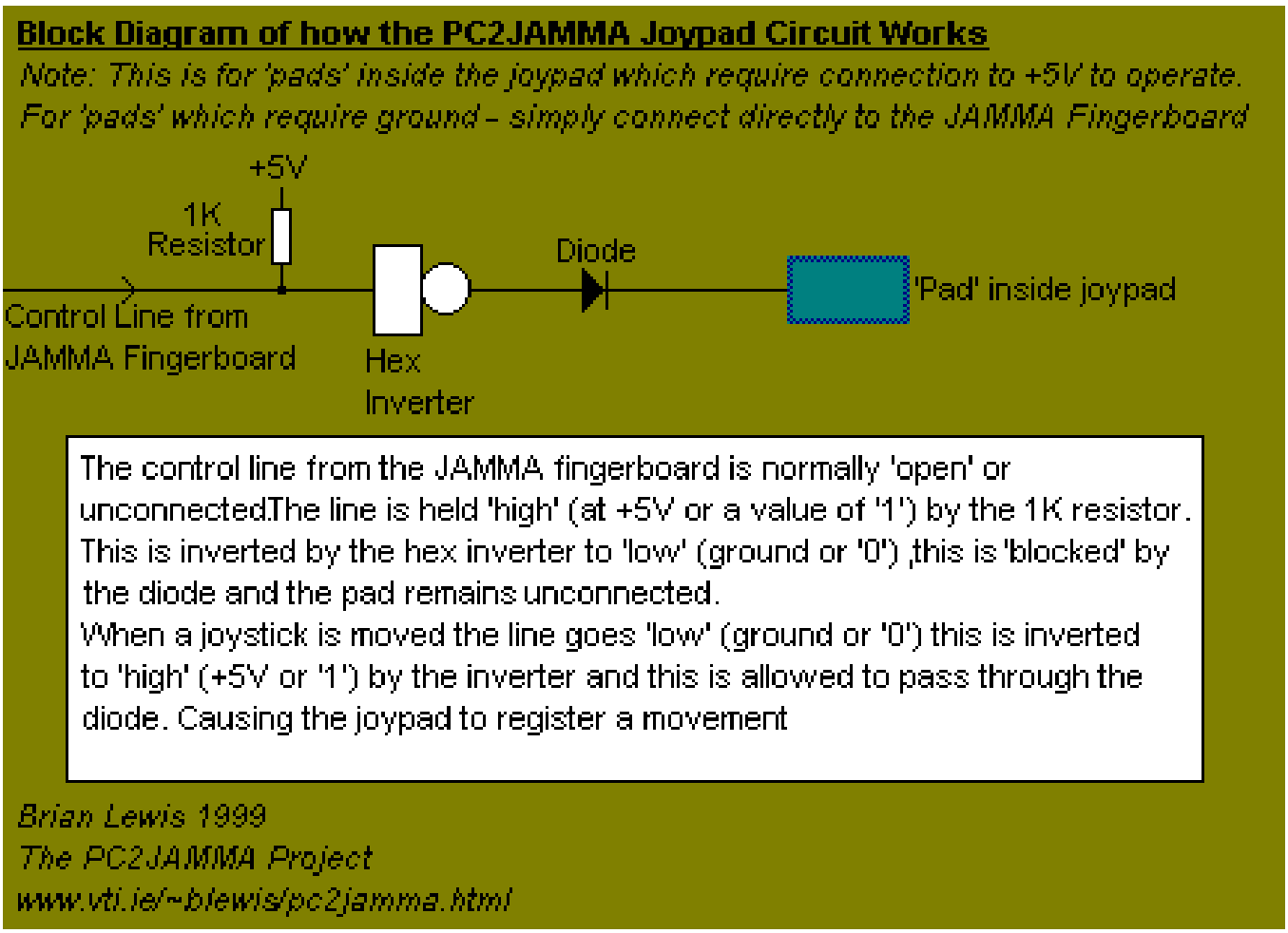
You'll need to power the circuit from the keyboard's powersupply; otherwise you'll get 'keyboard stuck' errors when you turn the PC on. To identify power and ground you can either:-
Check the pinouts for you keyboard
and/or
check the voltage across a likely looking pair - one should be black, one red.

The Joypad.
In the end I didn't bother using a joypad; I simply wired in all controls from the keyboard - but using a joypad does actually cut down on the amount of hardware you need to make.

Open you your joypad, you'll see that it consists of pairs of 'pads', these pads are shorted together when a button is pressed or the D-Pad is moved.
One side of the pad will be 'common' to all or some of the controls,
you'll need to identify which pad this is.
In the joypad I had , the 4 buttons had a common ground pad, and the 4 directions had a common 5V pad. So, wiring in this particular joypad means you need no hardware whatsoever to wire the buttons to the JAMMA fingerboard. Simply solder a wire from the 'non common' pair of each button pad and solder the other end to the correct pin on the JAMMA fingerboard.
(The ground will already be common because of the video)
So when a button is pressed in the cabinet - it'll complete the circuit for the button 'pad' on the joypad.

The directions were a bit more tricky, as it needed 5V rather than ground connected to it. The idea again is to take a control line from the JAMMA fingerboard, hold it high (5V) with a 1K resistor then run it through an inverter (7404). This will invert the signal correctly, but will leave ground connected to the pad when there is no joystick movement. So we need a diode between the pad and the inverter - to make sure only +5V get's through.

The circuit for one control line is as follows
(Click on it for more legible version)



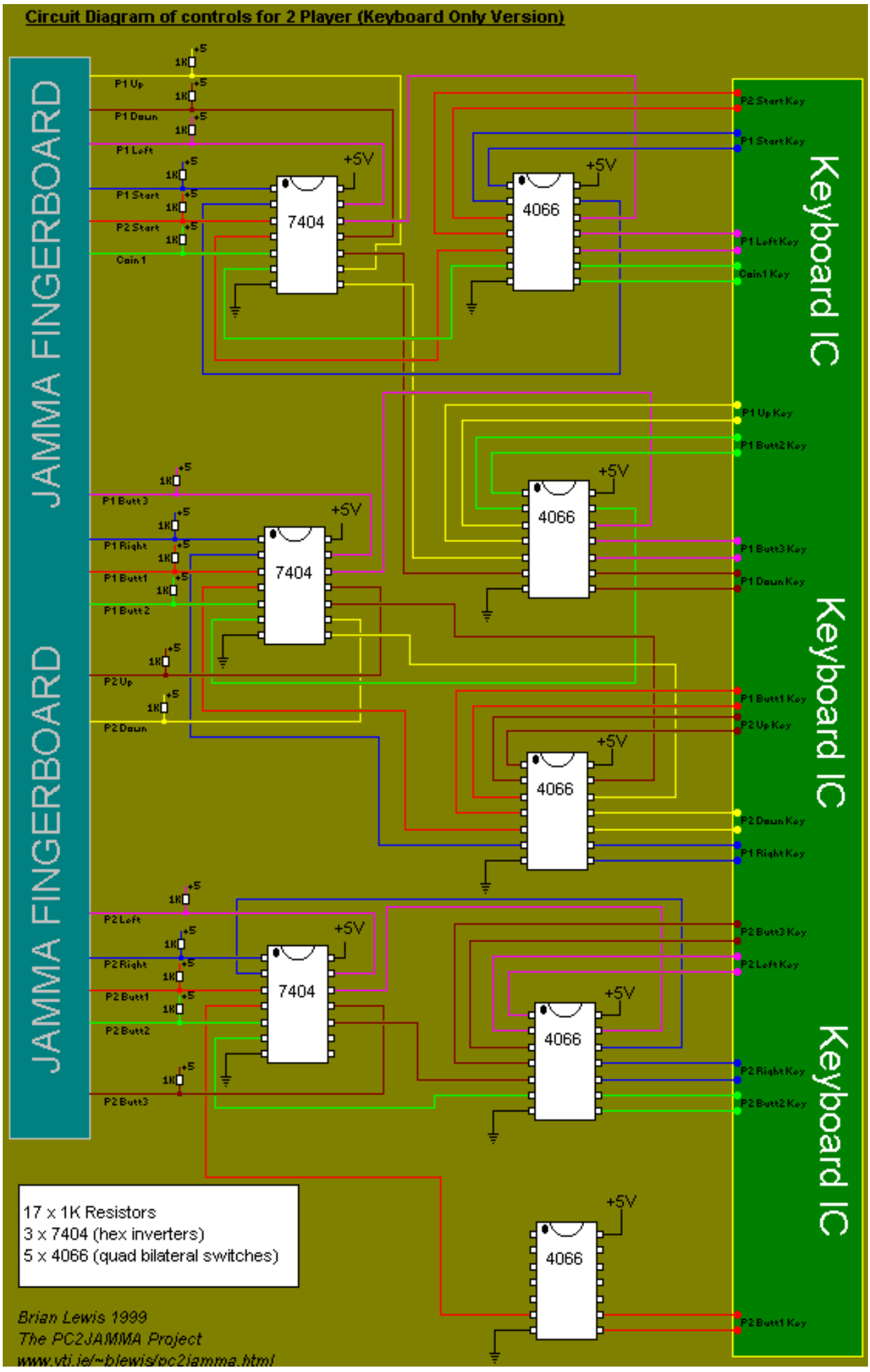
You can see what it a circuit for the 4 directions looks like
made up on a bit of stripboard here

Full circuit diagrams for the PC2JAMMA wiring options are given on the next few pages

Controls

2 Player, 3 Button Version Using just the Keyboard
This is the setup I use

Components
17 x 1K Resistors
3 x 7404 (Hex Inverters)
5 x 4066 (Quad Bilateral Switches)
8 x 14pin DIL Sockets
The complete circuit diagram looks like this
(Click for a more legible version)


Circuit Diagram of controls for 2 Player (Keyboard Only Version)
JAMMA FINGERBOARD
JAMMA FINGERBOARD
P1 Up
P1 Down
P1 Left
P1 Start
P2 Start
Coin 1
P1 Butt3
P1 Right
P1 Butt1
P1 Butt2
P2 Up
P2 Down
P2 Left
P2 Right
P2 Butt1
P2 Butt2
P2 Butt3
1K
+5
+5V
7404
4066
Keyboard IC
P2 Start Key
P1 Start Key
P1 Left Key
Coin1 Key
P1 Up Key
P1 Butt2 Key
P1 Butt3 Key
P1 Down Key
P1 Butt1 Key
P2 Up Key
P2 Down Key
P1 Right Key
P2 Butt3 Key
P2 Left Key
P2 Right Key
P2 Butt2 Key
P2 Butt1 Key
17 x 1K Resistors
3 x 7404 (hex inverters)
5 x 4066 (quad bilateral switches)
Brian Lewis 1999
The PC2JAMMA Project
www.vti.ie/~blewis/pc2jamma.html

Electronics Free Version
If you don't fancy making up the control line circuits, you can simply rewire your cabinet.
Take each pair of wires from your keyboard/joypad and simply solder them onto the appropriate switch inside your cabinet.
By the same token, you don't have to make the audio amp. yourself, you can either buy one (complete or kit) or mount PC speakers in your cabinet

Rewiring your cabinet in this way is at least as much (and probably more) work than the methods outlined previously, and it also ruins the whole PC2JAMMA idea - but it is an option.
It just means that rather than having a PC you can put in **any** JAMMA cabinet;
you end up with a cabinet that **only** a PC can go into.

In Use

Once you've got your adapter together and you've tested it, (You can test the controls/amp outside of the cabinet, and test the video by running MAME with the **-NTSC** option, then turning off /unplugging the monitor , plug in the VGA lead you've made then turn on the arcade monitor)
you can set up your PC to run inside your cabinet.

I recommend you don't used Win95, Win98 on the target machine.
It has a large footprint, long startup time and a nasty habit of 'disk thrashing' which ruins your frame rate.
Not to mention the fact that it could accidentally 'jump' into a high res video mode (which may damage your monitor).
Use DOS instead, with an appropriate DPMI server (CSDPMI3B.EXE)
If you have Win95/98 on the target machine and don't want to uninstall it, simply edit the file **MSDOS.SYS** in your boot directory. This is a read-only, hidden, system file; so you'll need to reset these attributes before you edit it

attrib msdos.sys -s -r -h

then edit the file, and change the line

BootGUI=1

to

BootGUI=0

This will stop the Win95 GUI from coming up and just leave you in DOS mode.

If you still want to run under Win95 you might find CLOSEWIN.EXE useful. It's a small program I wrote to close down windows. You can place it at the end of your batch file.

If running in DOS, simply edit your autoexec.bat to run ArcadeOS.EXE

i.e.
cd \ArcadeOS
ArcadeOS

If you're running in windows you need to create a batchfile, then place this in your StartUp folder.

Make sure you read **ArcadeOS.DOC**, that your emulator setup is compatible with ArcadeOS.EXE and that you have ArcadeOS.CFG setup correctly.
(most important thing being **incabinet = 1** is set in ArcadeOS.CFG when the PC is connected to an arcade monitor)

Once everything is setup , turn on your PC; wait for the 3 beeps, then turn on your cabinet.
Then (Finally) - play some arcade games.

Controlling the Cabinet/Monitor from the PC

What follows are instructions to build two circuits, these connect to the parallel printer port of your PC and either :-

a)Cut signals to your arcade monitor until ArcadeOS has set your video card into a safe scanrate/resolution

b)Turn on your arcade cabinet after ArcadeOS has set your video card to a safe scanrate/resolution

This is a 'finishing touch' to the project and **neither** circuit needs to be built in order to make the PC2JAMMA project, and you can skip past this section if you don't want to build either circuit.

It is simply to get round the 'boot problem'
(i.e. your PC boots up using a video mode which cannot be displayed on an arcade monitor.)
ArcadeOS gets round this problem in one of two ways :-
1) It beeps 3 times when it has set a safe scanrate/resolution -
so you know it is safe to turn your arcade cabinet on
2) It sends a signal to the parallel printer port to operate one of
the circuits described here.
You do **NOT** need to make either of the following circuits -
you can simply turn your PC on, wait for the beeps then turn your arcade cabinet on.

A Word of Warning

There are a few reasons while these circuits took a while to appear on this site.
One is the fact that it's the last thing I built,
another is the fact that I used 'equivalent' components to make the circuit. (see text)
However, the main reason I was reluctant to put the circuits up was the fact that one or both of these circuits are potentially dangerous to make and use.
The circuits described here differ in two ways from the others on this site:-
1) The circuit is connected DIRECTLY to the PC rather than through a soundcard or keyboard - greatly increasing the chances of damage to the PC itself, if you make a mistake in constructing the circuit.
2) The 'Relay Version' of the circuit involves AC mains - which greatly increases the chances of personal injury and/or damage to equipment
If you do not have any previous experience with electronics I strongly advise you **AGAINST** making the Relay circuit.
If you still want to make one of these circuits go for the 'cut monitor signals' version instead.

Controlling the Cabinet/Monitor from the PC

The PC Parallel Printer Port
Both of these circuits connect to the parallel printer port of a PC, so we'll start with a quick overview of how that port works.

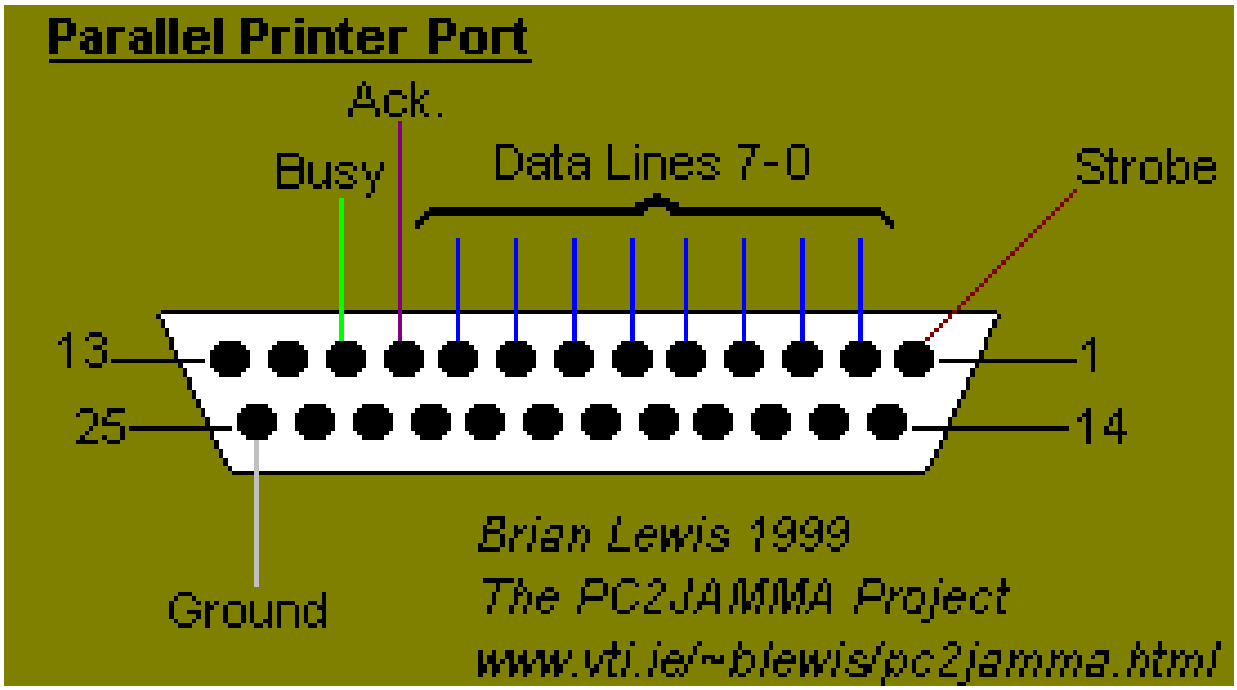
It's probably the easiest port on a PC to understand and make hardware for.

The operation of the port (from the PC end) is as follows:-

The byte to be sent to the printer is setup on the 8 data lines of the port.
+5V for 1 , Ground or 0V for 0
The PC then sets the 'Strobe' line to 1 (+5V), the byte is read by the printer and a 'Busy' flag is sent back to the PC.
Once the byte has been processed by the printer it sets the 'Busy' line to 0 (0V or Ground) and the 'Acknowledge' line to 1 (+5V).
The process repeats itself for every character sent to the printer.

Since each of the bits of the byte sent are on physically separate wires,
as are the 'Strobe', 'Acknowledge' and 'Busy' signals, it is fairly easy to make hardware which will appear like a printer to the PC, but can perform very different tasks.

The pinout of a PC's parallel port is as follows



To attach a circuit to the port, it needs to operate something like this:-

---

It will have a latch register connected to the 8 data lines of PC port. (This is like a simple memory chip - it has inputs which are not read until a line on the chip is set to high (+5v); the inputs are then read and copied to the outputs. These outputs will not change until the 'read line' is set to high again. This preserves the byte sent to our 'printer')

It will have a timer to generate the 'Acknowledge' and 'Busy' signals back to the PC.
(This will be kicked off by the 'Strobe' output from the PC. The delay really only has to be long enough to allow the latch register to operate)

Controlling the Cabinet/Monitor from the PC

The Circuits
Like most of the circuits on this site - the following two are made from components I had 'at hand'.
As such, neither contains a latch register or a timer.

Instead a 4510 counter is used to act like a latch register and a 7404 hex inverter is used to act like a timer/delay.
This isn't so bad - it cuts down on both the amount of soldering and the cost of the circuits. **Making a Latch Register from a 4510 Counter**
Many ICs can be made to act like a latch register, in the case of the 4510 counter we can use its weighted inputs and outputs and its 'parallel load' line to make it behave like a 4 line latch register.

Basically, we attach one of the weighted inputs to a data line from the PC's printer port , the 'parallel load' to the strobe line and use the corresponding weighted output as our data 'latch'

When the PC sends a byte through the printer port,the counter 'loads' whatever is on the 4 input lines and these appear at the 4 outputs - and remain there until another byte is sent from the PC.

We're only interested in one 'bit' from the PC (data line 7) so we only need to wire in that input and the 'parallel load' line. All other inputs are made 'inactive' by connecting them to ground. (apart from 'Carry In' which ,on the 4510, uses inverted logic and must be connected to logic 1 (+5V) to be inactive)

Making a Delay from a 7404 Hex Inverter
The 4510 is handling our data input - we now need to take care of the 'Busy' and 'Acknowledge' signals sent back to the PC.

This is fairly simple, when the PC sends a 1 down the 'Strobe' line, we need to send a 1 back to the PC down the 'Busy' line. Then ,after a short delay, we need to send 0 down the 'Busy' line and 1 down the 'Acknowledge' line.

To achieve this we run the 'Strobe' line through a capacitor and a resistor (connected to ground) and then into one of the inputs on the hex inverter. The inverted output of this will become our 'Acknowledge' line, we'll then invert this output again to get our 'Busy' line.

What happen is this:-
The 'Strobe' line is normally held at logic 0 (Ground) and our 'Acknowledge' line (which is just the inverse of this) will be held at logic 1 (+5V). Our 'Busy' line is just the inverse of our 'Acknowledge' line, so this will normally be at logic 0.

When the 'Strobe' line goes to logic 1 , our 'Acknowledge' line goes to 0 and our 'Busy' line goes to logic 1. However, the capacitor attached to the hex inverter begins to charge (at a rate governed by the resistor). As it charges, less and less current flows through it - until eventually the input to our hex inverter goes to logic 0 (even though the 'Strobe' line is still at logic 1). This causes our 'Busy' line to go to logic 0 and our 'Acknowledge' line to go to logic 1.

So our interface receives the 'Strobe' input then sets the 'Busy' line to 1 and the 'Acknowledge' line to 0. Then, after a delay governed by the resistor and capacitor, sets the 'Busy' line to 0 and the 'Acknowledge' line to 1. Exactly duplicating the actions of a printer.

The resistor and capacitor used in the circuit produce a delay larger than is needed for the circuit to operate - this is just so you can see that the circuit is actually working by looking at its LEDs.

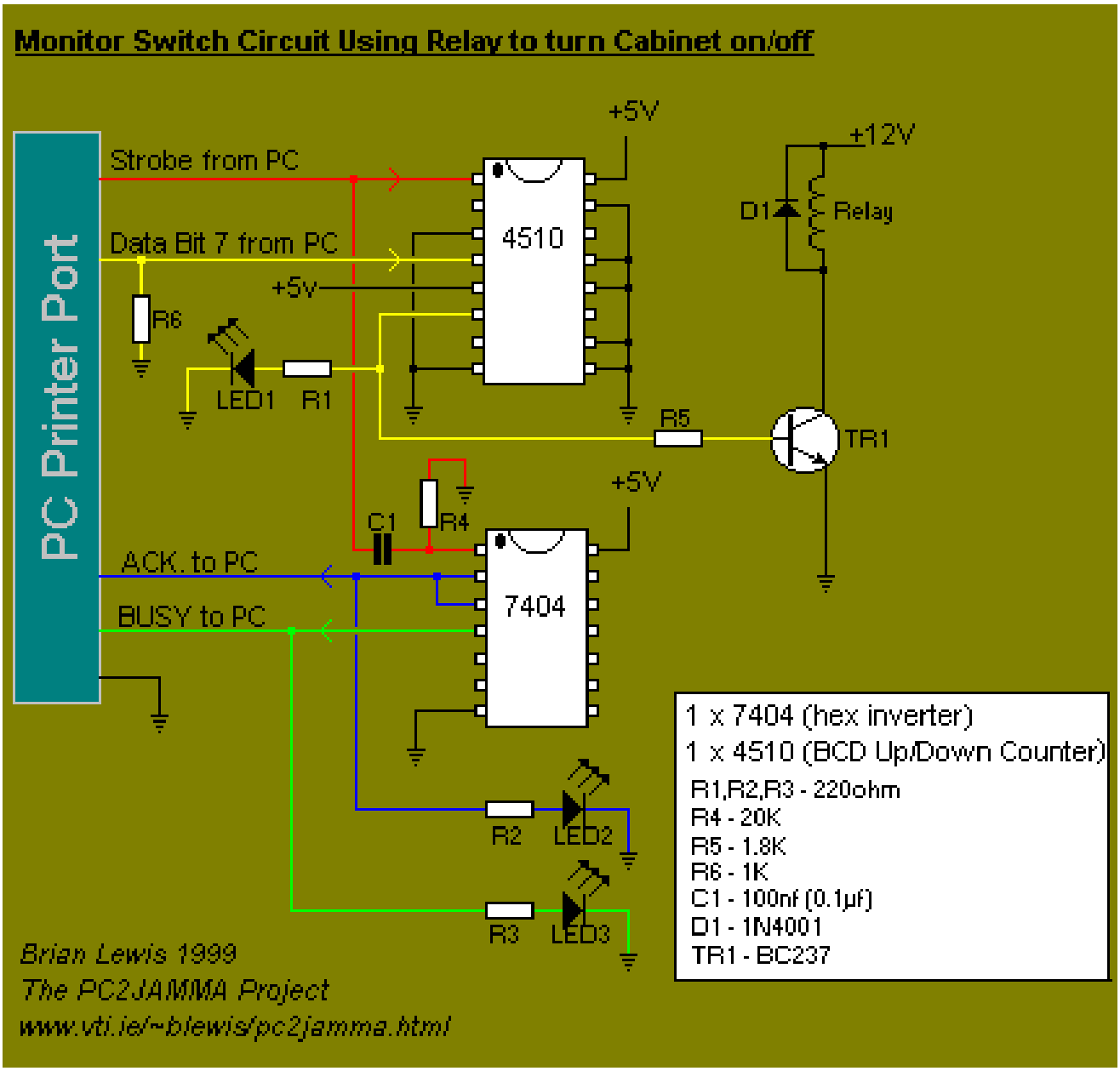
Controlling the Cabinet/Monitor from the PC

What You Need
For the Relay version of the circuit (which turns the arcade cabinet on and off)

1 x Parallel Printer Cable
3 x LEDs (pref. different colours)
3 x 220ohm Resistors
1 x 1.8K Resistor
1 x 20K Resistor
1 x 100nf (0.1 uf) capacitor
1 x Diode (1N4001)
1 x BC237 Transistor
1 x Relay (Coil - 12V, Relay - capable of carry mains voltage)
1 x 4510 (BCD Up/Down Counter)
1 x 7404 (Hex Inverter)
1 x 14pin DIL IC Sockets
1 x 16pin DIL IC Sockets

2 x Double Throw Switches (for testing)

The circuit diagram is as follows:-
(Click for a larger version)



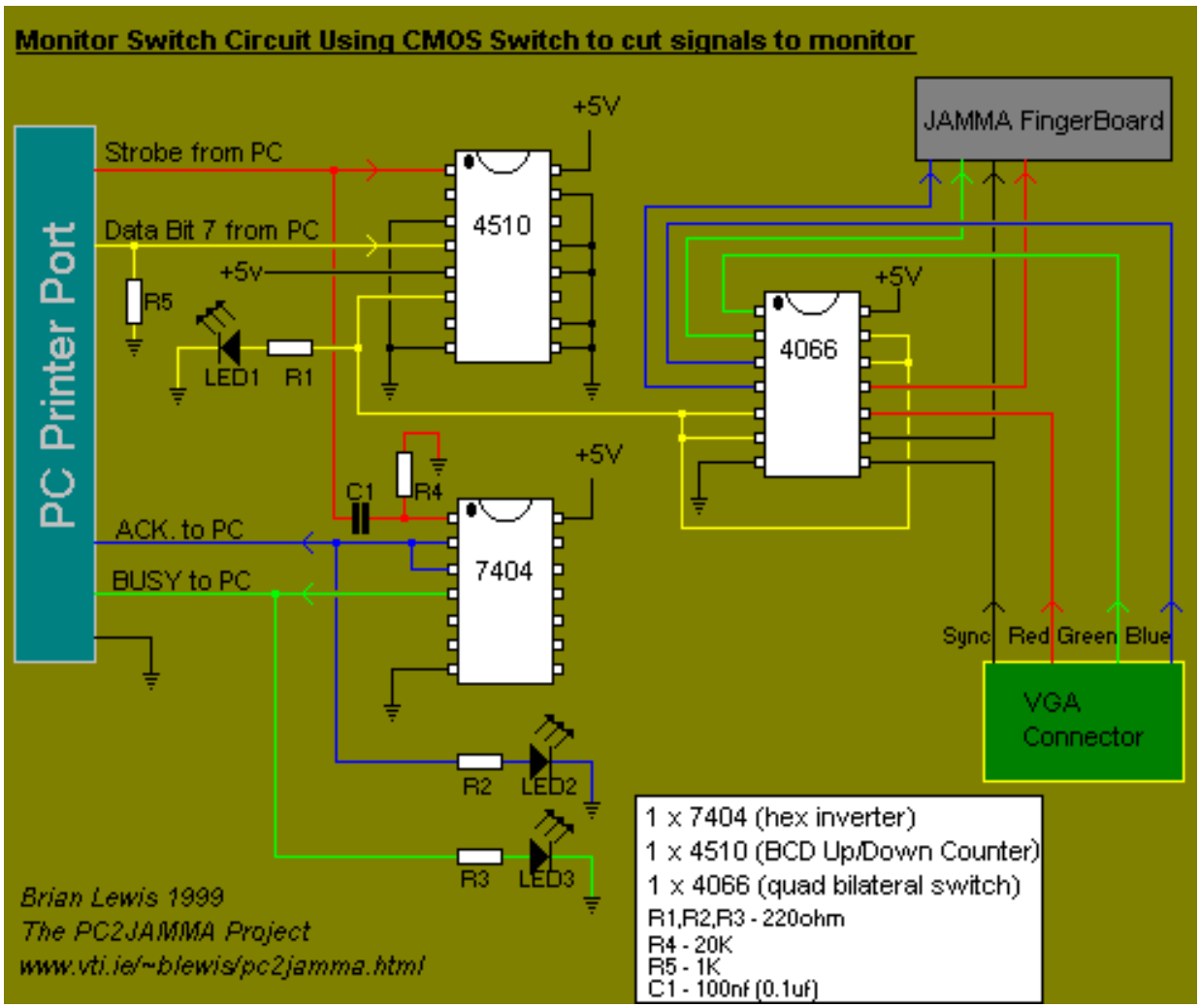
For the 4066 switch version of the circuit (the safer option - simply cuts the signals to the monitor)

1 x Parallel Printer Cable
3 x LEDs (pref. different colours)
3 x 220ohm Resistors
1 x 20K Resistor
1 x 100nf (0.1 uf) capacitor
1 x 4066 (Quad Bilateral Switch)
1 x 4510 (BCD Up/Down Counter)
1 x 7404 (Hex Inverter)
2 x 14pin DIL IC Sockets
1 x 16pin DIL IC Sockets
2 x Double Throw Switches (for testing)
The circuit diagram is as follows:-
(Click for a larger version)

Note: When you buy the printer cable make sure that the plugs are screwed together rather than sealed. This make identifying which wire goes to which pin a lot easier - as you can just open it up and look. Also, you can by a 25pin Serial Cable instead if you wish (it's what I bought for this project).
They're cheaper and have significantly less wires so are easier to work with. HOWEVER , not all the pins we need will be connected - so you will need to rewire the socket for this project.

To use your cable in this project, simply cut the cable (leaving a good length attached to the male socket) then open up the socket and identify the wires you need:-

'Strobe' (pin 1)
'Data bit 7' (pin 9)
'Acknowledge' (pin 10)
'Busy' (pin 11)
'Ground' (pin 25)



Controlling the Cabinet/Monitor from the PC

Building the Circuit
As the circuit is probably the most operationally complex on the site, I recommend you build it in stages and test each stage.

Start by just wiring in the 4510, R1 and LED1, then test if this is working correctly.

Do this by 'pretending' to be the PC printer port. Wire in 2 double throw switches, one for the 'Strobe' input to the 4510 and one for the 'Data Bit' input to 4510. With one side of each switch connected to Ground and the other side of each switch connected to +5V. So you can set the input to 1 or 0 for both the 'Strobe' and 'Data' lines. Set both switches to '0' and power the board from a bench power supply (or a mains power adapter capable of generating 5V to 6V DC)

When you power up the circuit with both switches set to '0', the LED should be unlit.

Now, set the 'Data' switch to '1'
The LED should remain unlit
Then flick the 'Strobe' switch to 1.
The LED should now light.
Now flick the 'Strobe' switch back to 0.
The LED should still remain lit.

Now flick the 'Data' switch to 0
The LED should still remain lit.
Then flick the 'Strobe' switch to 1.
The LED should now be unlit.
Flick the 'Strobe' switch back to 0.
The LED should remain unlit.

Once you've got your 'latch register' working as described above, you should add :-

the 7404, C1 , R2 , R3 , R4 , LED2 and LED3

to the circuit to get your Busy/Acknowledge lines working.
Now perform the same test you did for just the 4510.

Set both switches to 0 and power the board.

The 'Data' LED (LED1) should be unlit,
The 'Acknowledge' LED (LED2) should be lit,
the 'Busy' LED (LED3) should be unlit,

As you perform the previously described test,
the 'Data' LED (LED1) should behave as before,
in addition the 'Busy' LED (LED3) should flash very briefly each time the 'Strobe' switch is set to '1'.
You probably won't be able to see the 'Acknowledge' LED going off when the 'Busy' LED flashes - as the delay is too short - but if the 'Busy' LED flashed then it's a safe bet that everything is working.

Now remove your test switches and wire in the printer cable.

You need to wire in:-
'Strobe' (pin 1)
'Data bit 7' (pin 9)
'Acknowledge' (pin 12)
'Busy' (pin 11)
'Ground' (pin 25)

(The port's Ground just needs to be connected to any ground on the circuit)

Then download 'MONTEST.EXE' from the downloads page so you can easily test the circuit when it's connected to a PC. Plug your printer cable in , power the circuit and run MONTEST.

If everything seems to be working - finally add the Relay or 4066 switch to your circuit (depending on which circuit you're building).

Then test again using MONTEST.

Testing the relay is fairly easy as you will hear it click as it switches on and off. To test the 4066 switch you need to get a multimeter and check the resistance between one set of 'switch pins' (say pins 1 and 2). When the switch is 'open' the resistance should be infinite, when it's 'closed' it should be about 100 ohms.

Now you just need to wire it into the rest of the project.

If you've made the 4066 switch version of the circuit, you can power it either from the PC's power supply or your arcade cabinet. The relay version must , obviously, be powered from the PC.

For the 4066 version simply run the R,G,B and Sync lines through the 4066 as indicated in the circuit diagram.

For the Relay version you need to run the AC mains powering the cabinet through the relay

(Once again - if you have no previous electronics experience I advise you **AGAINST** making the relay circuit.
A mistake in construction could result in a ruined computer - or even ***electrocution*****)**

Once everything is in place set

printhardware=1

in **ArcadeOS.CFG**

Now, when you run ArcadeOS it will not allow signals to your monitor/turn on your cabinet until your video card is running in the right mode.

Alternatives to using a real Arcade Monitor

This project is meant to provide a (cheap) solution to the problem of putting a PC into an arcade cabinet. I approached it like the other arcade projects on this site - with the idea of making a harness to sit between the PC and the cabinet, without alteration to either.

It was also meant to provide a solution which was as accurate as possible -
There are other solutions to the 'monitor' question, I've outlined 4 of them below

.Replace the arcade monitor with a PC Monitor

This is the route most MAME cabinet projects take, and although it has its advantages (you can run anything inside your cabinet) It tends to lessen the 'arcade experience' and makes the project look like what it is - a PC sat inside an arcade Cabinet.

It also means that you'll probably have to run all your vertical games horizontally, so they'll look squashed and have large black borders down each side of the screen.

This is because most PC monitors are not designed to be rotated, and it can be dangerous to do so.

It's also expensive - you'll have to spend a fair amount of money to get a PC monitor the same size as a standard arcade monitor. Even then, the extra 'bulk' of the monitor's casing will mean you'll probably have to settle on a smaller screen than the cabinet's original monitor.

.Use a VGA2TV card or external convertor

Not a bad solution, but you'll have to be careful of 2 things
i) Make sure that the convertor you buy generates RGB and not just composite video. If it only generates composite video you'll have to either replace your arcade monitor with a TV or hunt around trying to find a (probably discontinued) TDA chip which separates composite video into RGB signals (plus a LM1881N to get your composite sync). Or you could try and find the separate RGB signals on your card/convertor before they're combined into a composite video signal and use those as your outputs. Converters which generate RGB are usually more expensive then those that just generate composite video.

ii)Make sure the convertor generates an interlaced image without dropping the frame rate in half. Interlacing is a technique used to generate a high resolution picture by drawing it in 2 passes. Converters generally store the image from a PC in a frame buffer and draw from that frame buffer for the 2 passes, odd fields on one pass, even on the other. However, the convertor may not update that buffer between the two passes - so 60 FPS suddenly drops to 30 FPS (if you buy a PAL/SECAM converter it could even drop to 25 FPS)

Note:The RGB signal produced will not be as clean as the one 'straight' out of the video card (using MAME's arcade monitor modes)

.Use software VGA2TV drivers which support your video card in hires modes

---

Once again you'll have to be careful about interlacing, not just a possible frame rate drop, but a vertical resolution drop. Some VGA2TV drivers create 'interlaced' hires pictures by not really interlacing at all - they simply only draw every other horizontal line.

You'll also have to only use 'real' VGA modes and SVGA modes, no 'tweaked modes'.

This is because VGA2TV drivers work by hooking into BIOS and trapping video mode change requests. When a video mode change is requested, the VGA2TV driver may call the original BIOS (to avoid having to setup all the registers) or simply setup all the registers 'manually' itself. In either case it will set the CRTC timing registers to values which will generate a TV compatible video output.

However, this is pretty much what 'tweaked modes' do as well. So if MAME wants to setup a VGA tweaked mode it will call BIOS to set the mode to 320x200 at 256 colours, the VGA2TV driver will trap the request then reprogram the registers for TV output. Then MAME will reprogram the registers again for the tweaked mode - effectively losing the VGA2TV driver's changes.

Note: ArcadeOS has a built in VGA2TV driver which get's round this problem by:
.Knowing before hand the tweak mode that will be set
.Allowing the emulator/app to set the tweaked mode and repogram the registers
- then reprogram the registers to an arcade monitor/TV compatible mode

.Replace the arcade monitor with one that supports VGA/SVGA inputs

---

This will still really just be a PC monitor - but it won't have bulky casing and should be the right size.
It'll allow you to run pretty much whatever you like in your cabinet.

You can check out a Wells Gardner Monitor that does the job here

The only downsides are that the display still looks like a PC monitor
and the cost.

There are other solutions to the ones listed, like using a MAC or Amiga instead of a PC etc.

Keyboard Clashes, Ghost Keys and CALLUS
When you're wiring in your keyboard you need to be sure that.
a)All keys work when any/all other keys are pressed
b)Multiple key presses do not generate false or ghost keys.

If you're just using MAME you should be pretty much okay,
(although left/up/alt together didn't work on my keyboard)
but there is a problem if you want to use the standard MAME keys with CALLUS.
If you're running CALLUS and you press the standard MAME button 3 (space) CALLUS' GUI appears, if 2 people are playing on CALLUS and player1 presses button2, and player2 presses button3 - CALLUS quits ('cause you just pressed Alt-Q)

My advice is to remap button2 and button3 to something other than Alt and Space (I use N and Z).
It's a pain to go through every single MAME game and remap the keys-
so I wrote a small utility (MAMESET.EXE) which allows you to remap the keys in all games.

Notes on 8Way / 4Way Joysticks"
A lot of older Games (PacMan,Ghosts n Goblins, Amidar etc.) had 4 way joysticks
in their cabinets. Which meant that it was impossible to do diagonal movements
in those games.

If you play these games using a modern 8Way joystick - you may come across problems
(such as being stuck on a ladder in GnG or going the wrong way in PacMan).

If you play the games 'cleanly' (i.e. don't don't use diagonals) it won't be a problem
but a better solution is to use an 8Way/4Way joystick in your cabinet
These are normal 8Way joysticks with a 'plate' or 'mask' on the back.
This mask is a square with rounded corners and can be rotated 90 degrees.
At one setting - it's a normal 8Way Joystick, at the other it's a 4Way.

I have one of these joysticks in my PC2JAMMA cabinet and it **DOES** make a difference
when playing the older 4Way games

Well that's basically it - you should now have enough information to make your own PC2JAMMA adaptor
Good luck.
