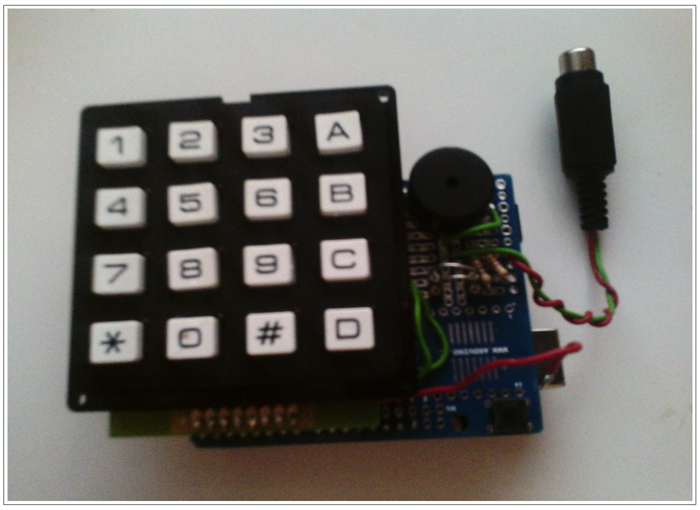
Project VIP

Introduction

ProjectVIP is a software and hardware based emulator for three 1802 based machines which are similar but slightly different – the Elf series, the RCA Cosmac VIP and the RCA Studio 2.



Each runs either as a PC based emulator running with the SDL library and developed under Code::Blocks or as a Hardware emulator running on an Arduino built using the Arduino IDE. The Hardware version generates composite video.

The system was developed and tested on Arch Linux x64 and an Arduino Duemilanove clone (effectively an Arduino Uno). It should work on Windows in both variants but I haven't tried this.

It should also work on an Arduino Mega2560 – this has the additional advantage that there is extra RAM memory available (the Uno/Due has a maximum of 1.5k, the Mega2560 should allow 6.5k).

Additionally using a Mega2560 will mean different pins are in use other than those specified here. However, apart from the TV interface pins, the Arduino version doesn't really mind which pins you use for the various parts.

This is an NTSC version, tested on an "Easycap" video grabber. It should work in PAL but I haven't tried it, and the system does not autodetect it. You have to change TV.begin(NTSC.....) to TV.begin(PAL ......) in the Arduino file.

License

As far as I am concerned people can do what they like with it – please credit my initial work if you do something else with it, or grab bits for other projects, or whatever. All the bits are here needed to make it work, it is basically a zip of my development tree.

Contact me

If you are having difficulties please feel free to email me at paul@robsons.org.uk.

The three machines supported

The emulators can only support one machine at a time – the system cannot flip between different systems like Emma02 or MESS. This is because the 'C' source has conditional includes for various bits of codes rather than having tests at run time.

Selection of the machine to be emulated is done by commenting or uncommenting the #defines in general.h – e.g. IS_COSMACVIP IS_ELF IS_STUDIO2. Only one should be included at once, the other two being commented out (it will not compile with two or more defined)

In general, the machines are slightly slower than the real ones, though only by about 5% or so. In practice, say playing a "Chip-8" game it is not noticeable.

There are other specific issues which are listed later on.

RCA Cosmac VIP

The Cosmac monitor has been modified so as to allow RAM in units of more than 1k. This should not affect the running of anything very much if at all.

Note that because the hardware emulator uses a standard keyboard which is available from most electronics outlets, which is similar but different to the Cosmacs.

1	2	3	A		1	2	3	C
4	5	6	B		4	5	6	D
7	8	9	C		7	8	9	E
*	0	#	D		A	0	B	F

The one on the left is the standard layout, the one on the right is the VIPs layout.

Pressing C and Reset boots into the Monitor as on a real VIP (though on the hardware as is it's actually "A", see above).

This is fudged slightly in the code so the upper bit of the address doesn't have to modified – it effectively jumps in at $800A once the latch has been reset.

RCA Studio 2

The system should run all ROMs that are available, but at present only supports one keypad – they are mapped on top of each other. It doesn't emulate the squawky sounding beep that a real machine has. Theoretically cartridges could have ROM all over the place and this is also not supported, though to my best knowledge all ROMs are between $0400 and $07FF.

The keyboard works as on a real machine, except A,B,C,D,* and # don't actually have any effect.

Elf

There are a variety of Elf systems. This one uses the keypad as laid out for the Cosmac VIP (there is no consistency in Elf keypads as far as I can see).

It misses out virtually all the basic Elf hardware – loading in code through DMA and In, there is no Memory Protect etc. The twin seven segment displays are emulated but are not supported in software or hardware.

The EF3 line checks a key press on the Elf keypad, one of many Elf keypad systems.

Others

There are other 1802/1861 derived machines which should be fairly easy to support if you want to – the Finnish Telmac machines for example.

The PC based emulator

Most of the development and testing was done on the PC based emulator in the CosmacVIP directory. This is built using Code:Blocks and SDL – so if you install CodeBlocks and SDL, create a new SDL project, copy the source files in there and just add them all in, it should work (famous last words).

It can be run from the command line and takes parameters of the form filename@address (e.g. ufo.ch8@200 to load file ufo.ch8 at address $200 hexadecimal). It starts in debug mode – you can change the code view address using 0-9A-F, data is the same with the shift key. P is reset, K sets a breakpoint, H sets the code pointer to R(P) , X sets the data pointer to R(X) , S single steps, G runs to breakpoint (interrupt with M) V "steps over" (SEP R3 type subroutine calls) and Escape exits.

Again you have to set the machine type in the general.h file.

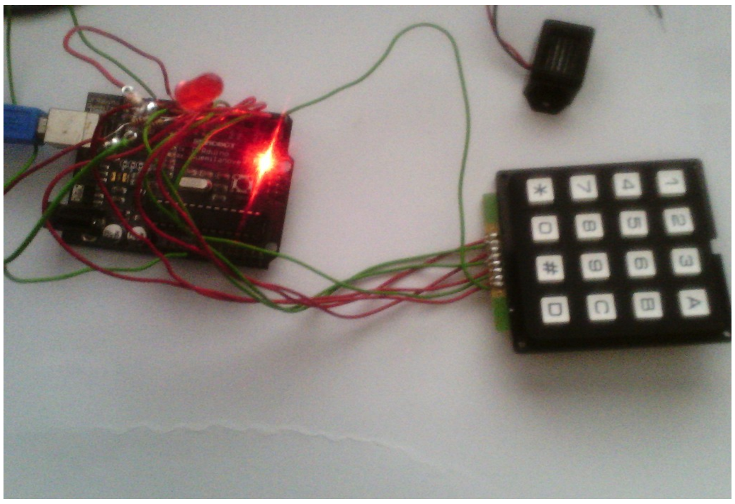
The Arduino based emulator

The Arduino emulator is somewhat more difficult – it has three parts

1) building the hardware
2) installing the Video generation library
3) building the app and running it.

Is there any easy way of delivering an "executable" in Arduino - just a binary blob to make it easier for people ?

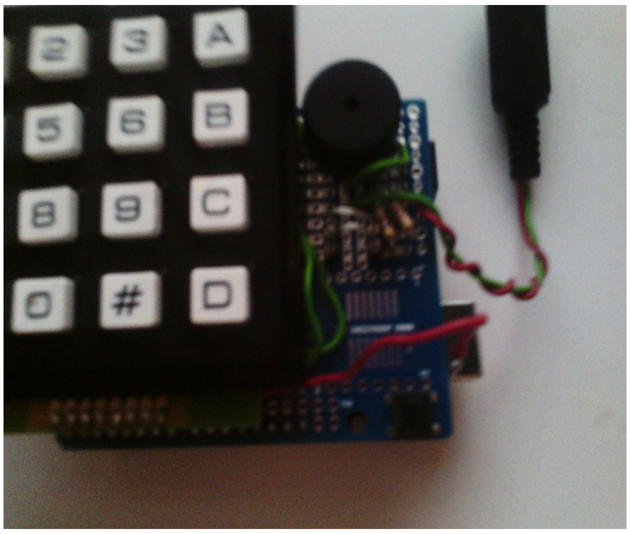
Hardware



The hardware is relatively straightforward and much the same for each machine. The main difference is that the Elf has an extra button (the IN button) and it's Q line is not connected to a beeper circuit (e.g. a circuit producing a fixed beep based on a 555 timer) but directly to the buzzer.

The picture is a prototype which doesn't have a buzzer but has a LED instead (see the picture on page 1).

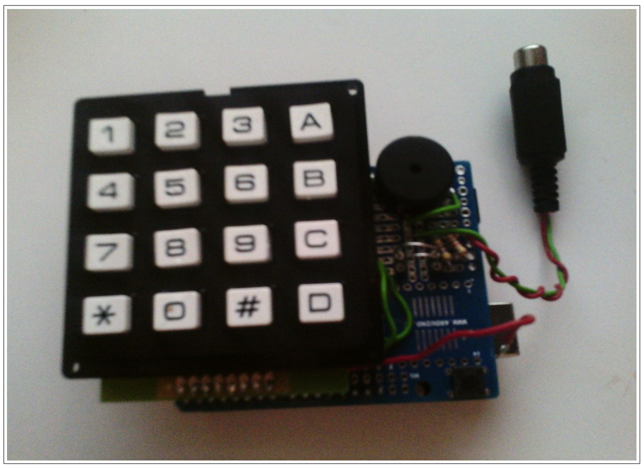
It is relatively simple to build – you need a 4x4 keypad, a piezo buzzer, a 470R resistor, a 1k resistor, a plug or socket to connect to composite video, and an extra push switch (for the Elf 2).



In my current version it is all built on an Arduino prototyping board (see picture).

There is no wiring under the keypad here – most of it is simple wiring to the buzzer and the resistors. The "floating red wire" is actually what I was using to test the "IN" button, but I presume anyone building one would use a switch.

You may notice there are two little wire loops, just below the piezo buzzer. These and the associated sockets are so if I test it on a Mega2560 (don't have one) I can wire it more easily, because the two mandatory connections are different on the composite video generation for a Mega2560.



The wiring is as follows (apart from the video this can all be changed).

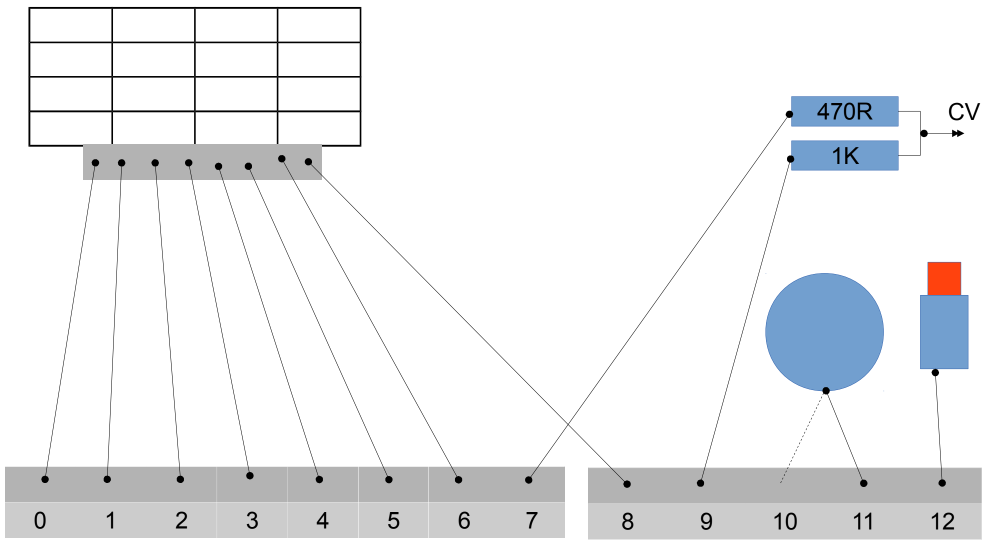
Keyboard : left to right 0,1,2,3,4,5,6,8. This fits conveniently with the layout because you can solder a 0.1" pin header into the keypad and solder that into the prototyping board – except for the rightmost pin which is connected to pin 8.

Video : connect a 470k resistor from the video pin (7) and a 1k resistor from the sync pin (9), join them together, this is the composite. (See *http://code.google.com/p/arduino-tvout/* for variations)

Audio : connect a piezo buzzer to either Pin 11 (Studio 2, Cosmac VIP) or Pin 10 (Elf 2). I really should fix this so they can all use the same pin.

IN Button : connect a button between pin 12 and ground. You only need this for the Elf, the others do not have it.

This is a rather pathetic attempt at a wiring diagram :)



The second connection to the piezo buzzer and the switch both go to GND.

Installing Libraries

The application uses a modified version of Myles Metzer's superb library "TVout".

Unfortunately the one thing it doesn't do well is 64x32 pixel graphics as used by all these machines – it occupies half the screen. I have therefore produce a tweaked version "TVout1802" which uses all of Myles' timing code, but has its own video generation code.

This is kept in the Arduino directory of the build tree and can be copied into the libraries directory of the Arduino install (in my machine /usr/share/arduino/libraries).

See http://arduino.cc/en/Guide/Libraries for more on libraries.

Building it

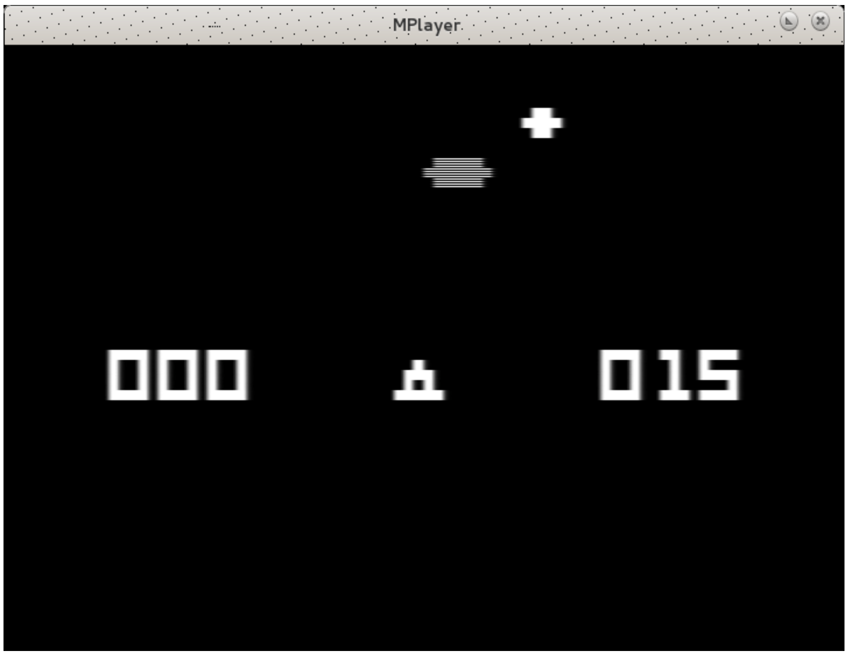
The best way to check the previous stages has worked is to start the Arduino IDE and look under Sketch/Import Library – this should contain a TVout1802 entry if the install has worked. Note if you had it running beforehand you will need to close and restart the IDE.

Now open the INO file in Arduino/vipntsc which is called vipntsc.ino, and it should compile and run.

Note: if you change any of the code relating to the emulator in the PC version it does not automatically propogate to the Arduino version – the Arduino IDE apparently cannot do cross-directory includes. There is a shell script "grabbit.sh" which copies the needed bits over (initially they will be there)

I'm sorry this is so messy but there doesn't appear to be an easy way of linking non-Arduino code into arduino sources.

Transferring binary data to a running Arduino application



Before you can get games or whatever running, you have to be able to upload them. The picture on the right shows it running UFO (the odd UFO colour is an artefact of the system).

I provide a program transmit.py in the Transmit directory. This works similarly on all machines (except at present it doesn't work on the Studio 2)

The program basically uploads binary files sequentially into RAM at $0000 (why it doesn't work on the S2 ahem :))

This application is in Python, so needs Python v2.7 plus pyserial. You will (on Windows) need to edit the serial port to use the Windows serial port (e.g. COM1, COM2) it is set to default to /dev/ttyUSB0 which is where Linux puts it. This is easy enough to do but I haven't tried it.

To upload images, run either

python2 transmit.py chip8.rom ufo.ch8 (for the Cosmac VIP)

or

python2 transmit.py speed.asm.bin (for the Elf 2, this is a test program)

while holding the lower right key down. This is either the D key (hardware keypad) or F (in reality) – on mine (see pictures) it's D.

When you hear a lower tone than the usual beep you can release the key, it should upload the images into RAM and run the machine – in the example above if would load chip8.rom into $000-$1FF then ufo.ch8 into $200 onwards.

However, this doesn't always work.

Sometimes, and I don't know why, it doesn't reset automatically – it is something to do with the Arduino IDE. In this case you have to manually reset it. e.g.

1. Press and hold D
2. Press the reset button and release it.
3. Let go of D when you hear the low beep
4. Run the python program as above

It's basically the same but slightly more fiddly. The DTR line is being disabled for some reason, this is why it doesnt auto reset.

However, it seems to operate in one mode or the other and work reliably enough. If you hear a higher beep and it appears to stop, then that is the checksum failing.

If nothing appears to be happening look at the pyserial documentation for more advice. http://pyserial.sourceforge.net/ I have had no problems with it (apart from the reset thing above which is the Arduino system somehow).

General issues

EF1 is not supported properly. It is jammed so that the 1802 doesn't go through the whole interrupt routine, and the Studio 2 ROM is modified to work with this.

Likewise, because of the non-emulation of the DMA on the 1861, it only works properly for 64x32 screens. “Souped up” 64x48 and 64x64 1861 won't work, though it is not difficult to change video_gen.cpp to make them work.

It was all developed on NTSC and not tested on PAL.

Paul Robson
paul@robsons.org.uk
18th March 2013.

First release : 18th March 2013