Grant's
sinclair
ZX80
Homebuilt hardware page

How to build your own ZX80/ZX81 (verified working)
and how it works.

...on-line since 1996 !

VERIFIED WORKING
RE-BUILT FROM INFO PROVIDED HERE

This page details the construction of your own ZX80. Follow this link to build an NMI generator which, when used with the ZX80 circuit detailed below and the ZX81 ROM, will produce a FULLY FUNCTIONING ZX81 USING OFF-THE-SHELF COMPONENTS.

by Grant Searle

For news and updates, follow me on Twitter:

Last update: 31st December 2016

Major changes log:
** 1996 - Page first shown on the Web*
** March 1997 - NMI (ZX81) conversion added to the Web*
** Nov 1997 - First (ever?) ZX80 software page added to the web*
** Jan 2009 - Accurately re-drawn schematic (vector graphics) - see it here (Adobe reader 7 or higher needed)*
** May 2009 - Version 2 of the NMI generator (exact timings) to convert the ZX80 into a ZX81 here*
** Jan 2010 - Accurately redrawn front and back foil patterns of the real ZX80 issue 2 board now available here (I believe I am the first and only one on the Net to do this!)*
** 14th May 2011 - Version 1.1 of the ZX80 issue 2 foils released here - see below for changes*

** 23rd May 2011 - Version 1.3 of the ZX80 issue 2 foils released here - see below for changes (1.2 was an interim version)*
** 28th May 2011 - Version 1.4 of the ZX80 issue 2 foils released here - see below for changes*
** 29th May 2011 - Version 1.5 of the ZX80 issue 2 foils released here - see below for changes*
** 3rd Jun 2011 - Version 1.6 of the ZX80 issue 2 foils released here - see below for changes*
** 25th Jun 2011 - Version 1.7 of the ZX80 issue 2 foils released here - see below for changes.*
Keyboard "asterisk" symbols updated in PDF.
My build of the ZX80 using PDF version 1.7 shown below.
** 27th Jun 2011 - Added circuit description and annotated circuit diagram here*
** 31st Jul 2011 - New version of the NMI Generator (ZX81 Conversion) released here*
** 31st Aug 2011 - New version of the NMI Generator (Zx81 Conversion) with exact ZX81 timings released here*
** 14th Sep 2011 - Detailed real oscilloscope pictures added to describe timing.*
** 15th Jul 2012 - Version 1.8 of the ZX80 issue 2 foils released here - see below for changes.*
** 19th Jul 2012 - Hole locations added as a text file within the foils zip - see below.*

Contents

INTRODUCTION

Nothing nowadays can replace the time when the first home microcomputers hit the streets. I am a keen collector of the home computer era. Follow this link to see my current collection. In those days you HAD to make your programs fit the hardware and memory available. This added to the enjoyment of programming to see something you had written to fit in a few K spring to life. The later ZX81 is very similar to the ZX80 except a lot of the ICs in the 80 has been merged into one chip. As a result, you can learn a lot about the workings of the ZX81 by referring to the ZX80 circuit. I have several computers from the start of the 80's but have always liked the Sinclair ZX80 for the following reasons:

1. It was the bare minimum required to produce a workable computer with a TV display.
2. It did not have a single dedicated IC in sight.

It is because of the second reason this page has come into existence. Here I show you how to build your own piece of history. Virtually all parts for the ZX80 are available from most of the larger electronic companies. The remaining parts (the Z80 processor and RAM, normally) can be picked up cheaply from eBay.

I must point out this is **not** a job for the absolute beginner. You will need to blow your own EPROM image and it will probably be a distinct advantage to have access to an oscilloscope in case it doesn't work. I can assure you the circuit supplied here as well as the PCB layouts DO work without any modification.

The details on this page consists of the original re-build done on strip-board, and also accurate reproduction foils to allow a faithful replica PCB to be made. Both versions have been built successfully so I know they are correct.

I made the following modifications for my stripboard version:
1. I originally used a 6264 SRAM to make the base memory 8K instead of 1K.
2. I have since replaced this with a 62256 (a 32K x 8 chip) to bring the memory up to the "full 16K" (the unused address line is tied to a power line).

Changes for both versions
1. I could not get the ceramic resonator. Instead I used a crystal and used a load capacitance of 47pF.

CIRCUIT DIAGRAM

I used this diagram to build my version of the ZX80. As a result, I know it is correct, apart from the following errors on the original schematic:

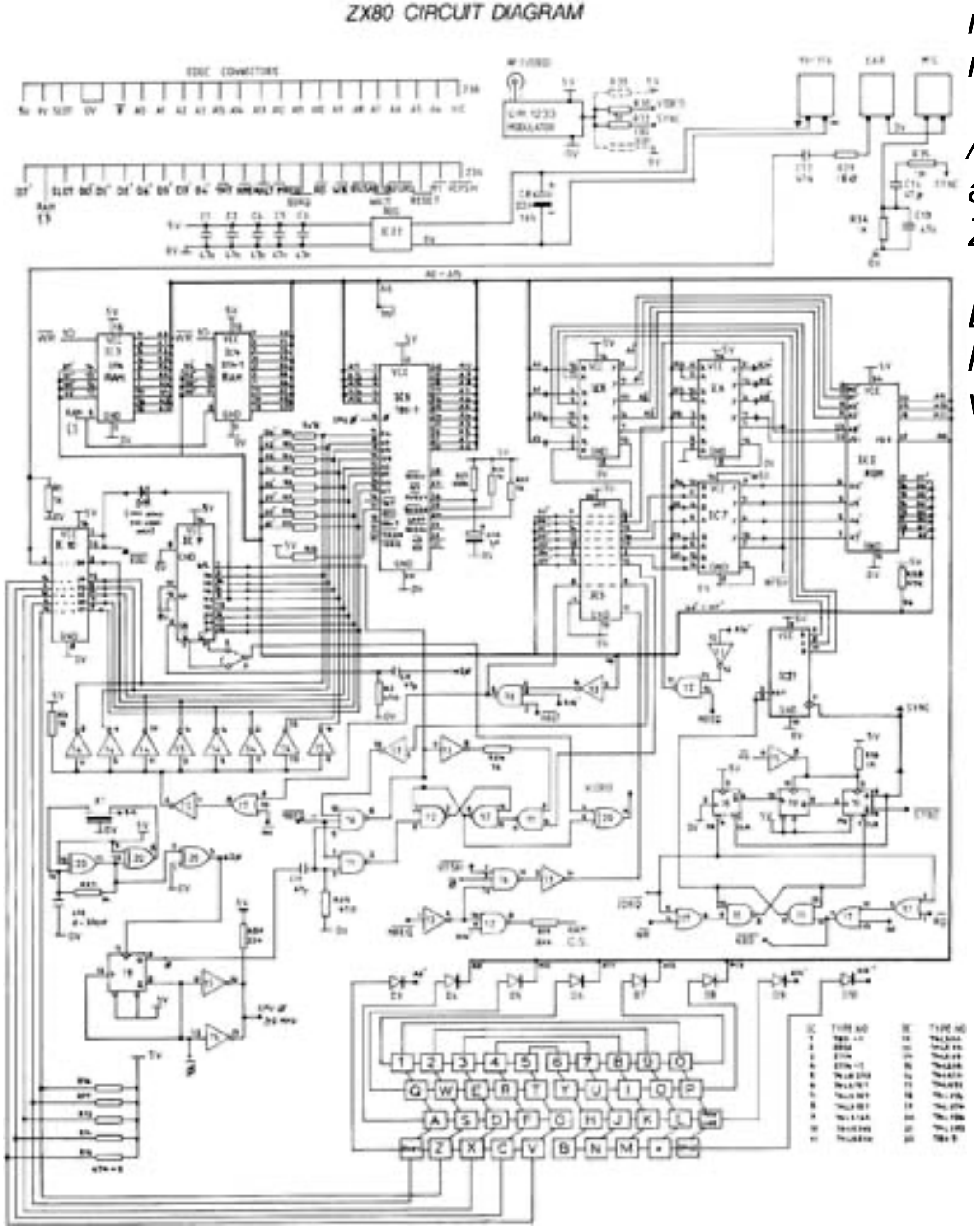
- R7 connecting D4' to D6. This should clearly be D6' to D6.
- R28 is shown connected to D6. This should be D6'.
- The Bus acknowledge signal on the CPU is /BUSAC, but it is labelled as /BUSAK on the edge connector.
- The component list shows IC9 as being 74LS166. This should be 74LS165, as shown on the actual circuit.
- One input on IC13 is marked as 1 but should be 11 (on the gate that has pin 10 as output).
- Parts list shows IC 11 to be 74LS500. This has been corrected to say 74LS00.
- Pins 4 and 5 on IC16 are swapped to match ZX80 PCB.

- Thanks to Ismael P�rez for finding many of the errors

Notes about the circuit diagram

MREQ is actually obtained from the output of the NOT gate, IC13,pin 2 (/MREQ is the input).

SYNC (IC19 pin 5) goes to the



microphone socket (via R35) and to the modulator (via R32).

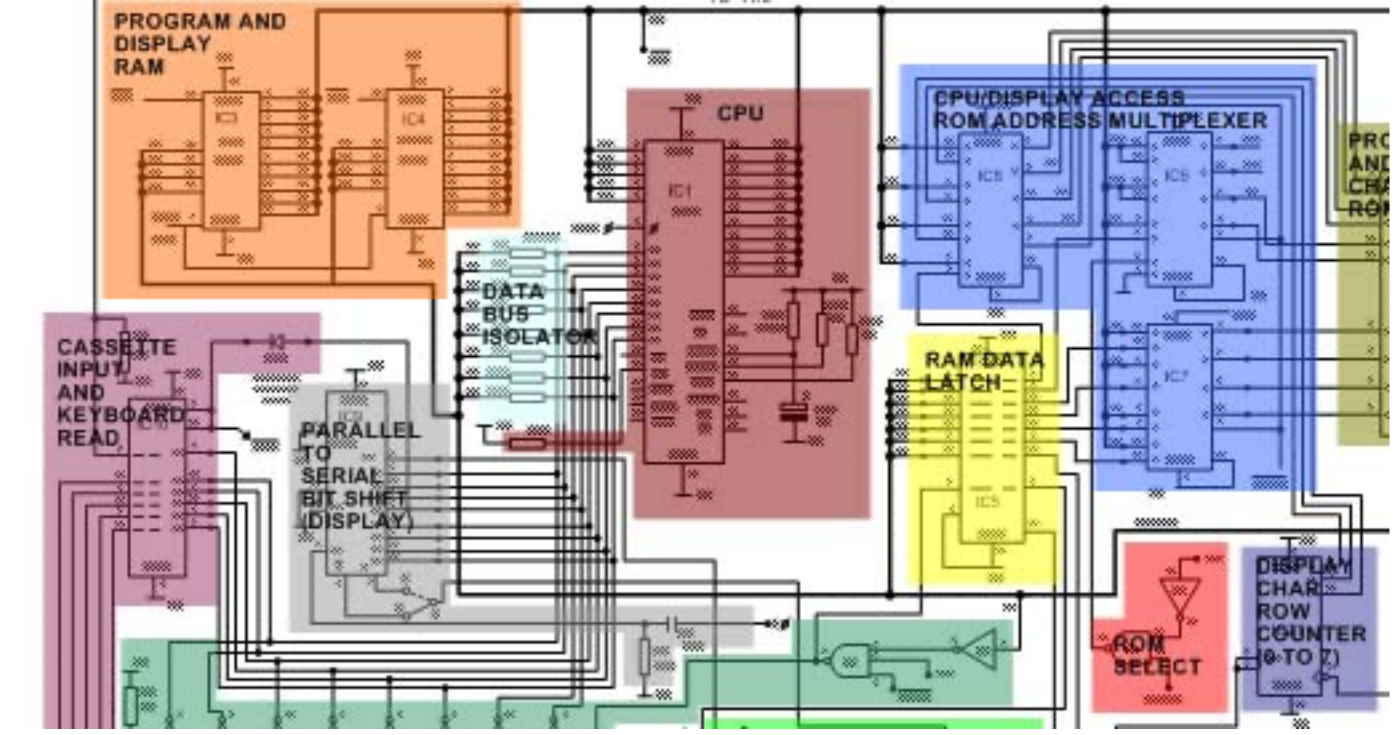
/SYNC (IC19 pin 6) is not actually used and goes to R36 (unused!) on the real ZX80 PCB.

Errors on the original scanned version have been corrected on my re-drawn version.

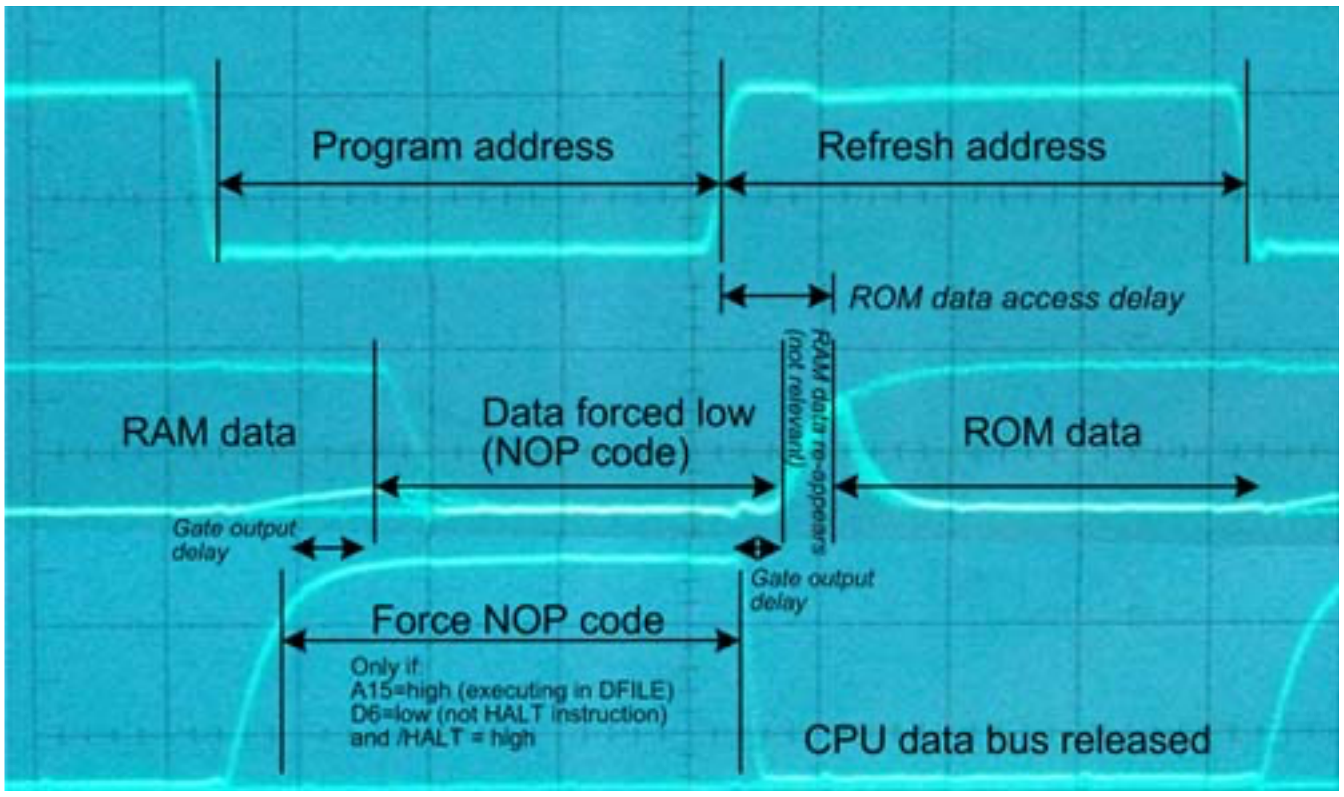
I have painstakingly redrawn this circuit using vector-based graphics so it can be scaled with no pixellation.
I have kept my version as faithful to the original circuit as is possible, but I have corrected the errors on the original schematic that I have mentioned above and I have added the missing pin numbers (4,10 and 13) on IC19 near the bottom-right of the scanned circuit. Apart from that, everything on my redrawn circuit is very accurately placed. I resized the scanned version to A3 and over-layed my new version on top of the scanned version while I was developing it. For most of the diagram, my placement is well within 0.5mm of the original when scaled to be the same overall size of the original. Adobe reader 7.0 or higher is required to open the PDF - you will get an error message if you use an older version.
Click here to get this re-drawn version (PDF format)
Click here to download the original scanned ZX80 circuit

CIRCUIT OPERATION

The following is a brief description of the various parts of the ZX80 circuit. Please refer to the colour-coded version of the circuit diagram (HERE, or click on the picture below) when reading these details.


PROGRAM AND DISPLAY RAM
CPU
CPU/DISPLAY ACCESS ROM ADDRESS MULTIPLEXER
DATA BUS ISOLATOR
CASSETTE INPUT AND KEYBOARD READ
PARALLEL TO SERIAL BIT SHIFT (DISPLAY)
RAM DATA LATCH
ROM SELECT
DISPLAY CHAR ROW COUNTER (0 TO 7)

In addition, to see detailed real oscilloscope traces and a description as to how the display is produced go here (or click the picture below).



Details of the ROM program operation can be found elsewhere on the net so will not be covered on this page. This text is intended to explain the functions of the schematic to explain the circuit operation when the ROM program is run, and can be used in conjunction with the explanation of the ROM code.

PROGRAM AND DISPLAY RAM
Static RAM holds both the program and also the display file (DFILE). The unexpanded ZX80 consists of 1K of RAM. This can be expanded to 16K directly addressable RAM and the internal RAM can be disabled via a line on the edge-connector.

PROGRAM AND CHARACTER ROM
First part contains the program ROM, the top part contains the character map. So, when the character data is being accessed, during the CPU refresh, the top address lines point to the start of a particular character, as determined by the character latch and the bottom three address lines are sourced from the display char row counter.

CLOCK GENERATOR
A ceramic resonator generates a 6.5MHz signal, which is then divided by 2 by IC18. The output is double-buffered to allow faster rise times on the CPU clock.
Other outputs are:
6.5MHz for the display bit-shifter
Inverted 3.25MHz clock to feed the character latch.

CPU
A Z80 processor running at an external clock of 3.25MHz. Unused inputs tied high, and the reset is connected to a simple RC circuit which holds the /RESET pin low for a short amount of time after power-up, ensuring the circuit is stable before the CPU starts execution. After reset, the address bus is zero, so the first program byte read is from the start of the program ROM.

DATA BUS ISOLATOR
The NOP generator will force "00" (NOP) onto the CPU data bus during display causing the CPU to run NOP code sequences while the address bus changes. However, RAM is also active during this time, so these resistors allow the CPU data bus to be forced low while the RAM is also outputting data.

CPU/DISPLAY ACCESS ROM ADDRESS MULTIPLEXER
This is controlled purely by the CPU refresh signal.
When refresh is high, the ROM address lines are connected to the CPU address, allowing normal program execution.
When the refresh signal is low, the ROM address lines are connected to the RAM data latch and the character row counter.

DATA LATCH ENABLE
Output low (RAM data latch will freeze outputs) when /REFRESH is low, /phi is low or /MREQ is high.
So, a normal memory read (or write), ie. /MREQ low and /phi high, will update the data in the latch with what is on the the data bus which is then frozen during the complete REFRESH cycle.

RAM DATA LATCH
The data latch enable circuit will allow IC5 to capture the data currently on the data bus, as read from the RAM. The lower 6 bits of this latched data is then used as part of the character address within the ROM. Bit 7 is the normal/invert display bit and bit 6 is used to determine whether the NOP opcode will be forced onto the data bus.
When the control input is high, the outputs follow the inputs. When the control input goes low, the outputs are frozen (latched) and will not change until the control goes high again.

"NOP" OPCODE GENERATOR
When using the CPU to draw the screen display, the CPU "executes" the start of the DFILE RAM addresses. During a program fetch cycle, if the data at the DFILE address has bit 6 low then IC 14 and 15 are active, forcing the data bus low (ie. settting it to be a NOP opcode). The CPU then executes this psuedo-instruction and increments the address bus to proceed to the next location. This continues until a byte is encountered with it's bit 6 high. In this case, the NOP code is not generated so the actual RAM contents is read and executed instead.

At this point, I should provide a note about the DFILE / NOP execution...
The display file (DFILE) is held in RAM and consists of 24 blocks of characters (one block for each line) terminated by a HALT instruction.
Every character has a code where bit 6 is zero, so the NOP circuit will be activated when the CPU executes that location.
At the end of each line of characters (which is variable length, between 0 and 32) is a HALT instruction, hex 76, binary 01110110. As you can see, the halt instruction has bit 6 set, so is executed as a normal instruction.
To display each line. the sync occurs, then a short delay. The refresh register is then loaded and interrupts enabled. The CPU then "executes" the start of the current line.
The refresh register (R) is preloaded with a suitable value so A6 will change from 1 to 0 once the end of the display scan is reached.
Each character is trapped and replaced by a NOP instruction on the CPU data bus while the display logic latches it and uses it as part of the ROM address that contains the character to be displayed. The ROM data is then serialised for display on the screen.
Once the HALT is executed, the CPU will then pause for the remainder of the line, where A6 eventually going low will then cause an interrupt to allow the CPU to continue with the next sync and line etc.
For more details, see here

RAM SELECT
The RAM CS signal is low (active) whenever a memory request is made (/MREQ=low) and address line 14 is high. RAM is therefore only partially address-decoded and is visible and repeated several

times (for the 1K machine) between 4000 and 7FFF, also between C000 and FFFF. The output is buffered by a resistor so this output can be forced to high/low via the edge connector.

ROM SELECT
The ROM select is active (low) when A14 is low, A12 is low and the MREQ signal from the CPU is also low. This partial decoding results in the 4K ROM being visible at 0000-0FFF, 2000-2FFF, 8000-8FFF and A000-AFFF. It is also enabled during the refresh cycle (as part of the display generation). Although the ROM is only ever read from, it is also active if the ROM is attempted to be written to. In this case, the ROM data and CPU data would be active (but isolated somewhat by the data bus isolator resistors). A15 is ignored, so the ROM is mirrored higher up in the address space.

DISPLAY CHAR ROW COUNTER (0 TO 7)
A 3-bit counter, reset on vertical sync, used to determine which of the 8 lines of a character is to be displayed.

PARALLEL TO SERIAL BIT SHIFT (DISPLAY)
IC 9 is a parallel-in, serial-out bit shifter. The data on the data pins is loaded via when the shift/load input goes low, and is shifted out, one bit at a time, by the clock (6.5MHz). This output data is passed to the display output logic. Normally pin 7 (/Q) is connected to the display, causing a black-on-white character to be displayed. If the PCB link is change to connect to pin 9 (Q) as shown by the dotted line on the schematic then the display will show white on black characters instead. C11/R25 ensures that only a short low pulse is present on the shift/load pin at the start of the phi clock being high.
The data is shifted out during the low-to-high transition of the clock pulse on pin 2 of IC 9.
The data is only loaded when the pseudo "NOP"s are being executed (ie. during the screen character drawing), as controlled by the data in on pin 8 of the data latch IC 5, and output on pin 9 of IC 5. During other periods of the display, no loading is done so the output pins of IC 9 remain static, producing the blank parts of the display (or the remainder of the current line in the DFILE).

NORMAL/INVERT CHARACTER LATCH
When a character is processed for display, data bit 7 (as captured in the RAM data latch) determines whether the character is to be inverse video. If this bit is set, the inverter latch is set, and the exclusive-or gate (IC20) will invert the data stream from the parallel to serial bit shift circuitry. This latch is reset at the end of a character display, unless the following character is also inverted.

I/O DECODER
/KBD is low when /IORQ is low, /RD is low and A0 is low.
So, the keyboard is read for every even-address I/O read. As a result, any peripherals requiring I/O reads would need to use odd addresses to avoid the keyboard read being activated.
In addition, any I/O write will set the latch implemented with IC11, and any keyboard read will reset the latch. This latch is used to generate the vertical SYNC signal.

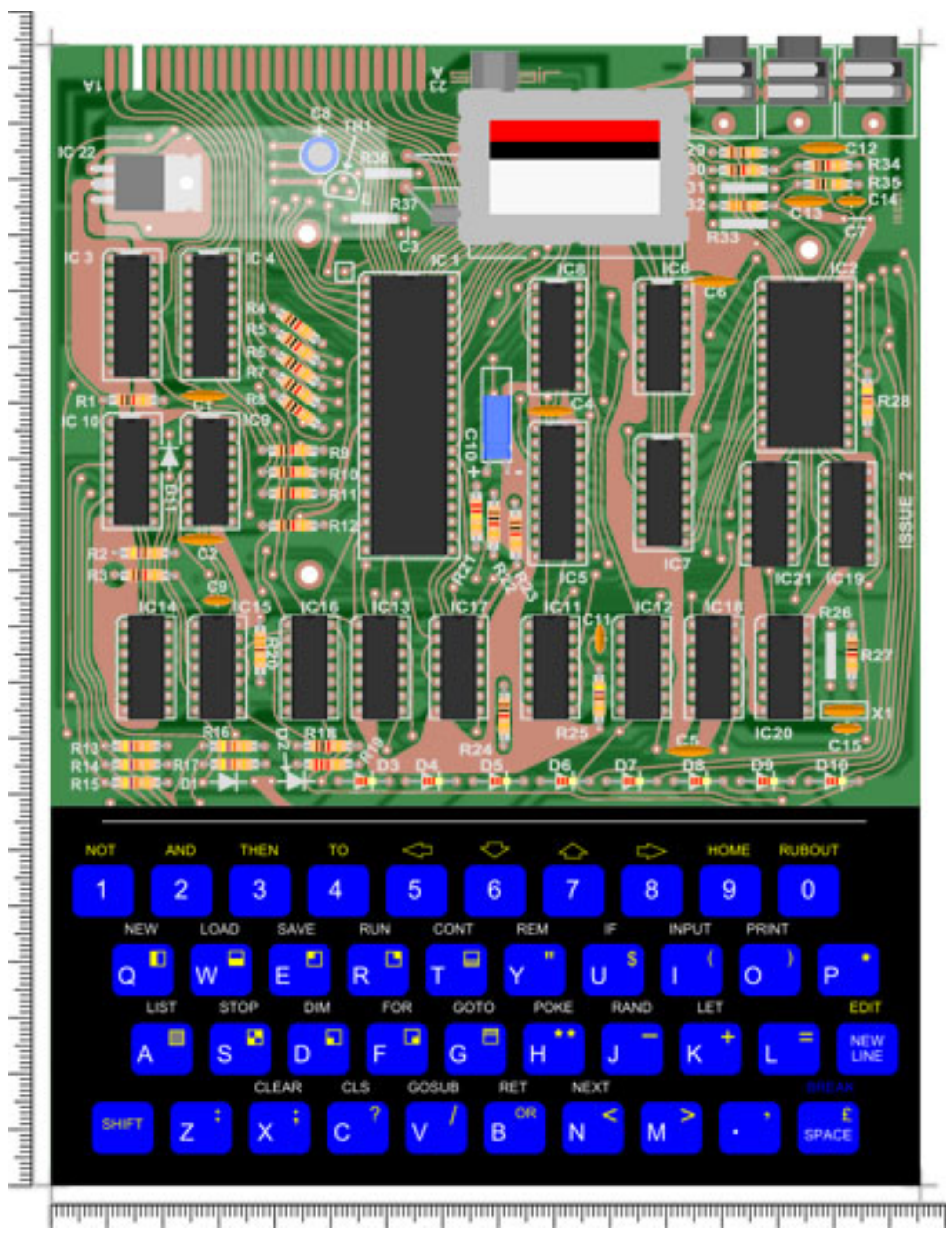
TV SYNC GENERATOR
Every INTACK (and I/O request) will preset the latch in IC 18. The display routine will halt the CPU until interrupted. It then generates an INTACK signal (/M1 and /IORQ low) which starts the sync generation. /M1 clocks the data through the two latches that eventually (14.5 cycles after IORQ going low) takes /SYNC low. The inverse (SYNC) output then resets the latch in IC 18 which is also clocked though this circuit, taking the /SYNC output high again after a short while. So, this circuit will generate a 20-cycle (6.15uS) low signal on the /SYNC line shortly after every I/O request. This 20-cycle sync is used for the horizontal syncs to the TV. In addition, the latch in the I/O decoder can force the output of the sync generator to be held low, as required for the longer vertical (frame) sync pulses on the /SYNC line.

CASSETTE INPUT AND KEYBOARD READ
The /KBD output from the I/O decoder is used to enable IC10, putting the data on it's inputs onto the CPU data bus.
The keyboard is read in a multiplexed manner, with only one of the address lines A8 to A15 low at

any particular time. If a key is pressed, then the corresponding data input to IC10 is pulled low, allowing the ROM to determine which key was pressed in the keyboard matrix.
Resistors R13 to R17 pull these inputs high when no key is pressed.
When the port is read, D4...D0 contain the keyboard switch details for a segment of the keyboard (as determined by which of A8 to A15 is low), D7 contains the cassette input value and D6 indicates whether UK 50Hz or USA 60Hz display is to be used (via D11).
D6 will be low for USA (due to D11 pulling D6 low when /KBD is low) and high for UK (R28 on the right-hand side of the schematic pulls D6 high if floating).

CASSETTE INTERFACE
Ear socket takes a signal, the capacitor blocks the DC content and the resistor limits the current when it is forced outside the normal TTL voltage levels.
Mic socket provides an attenuated (less than 1000th) signal that is compatible with the high-gain mic input on a cassette recorder. The mic input on a cassette recorder expects a very low level signal as provided by a dynamic (moving coil) microphone, so the attenuator provides a compatible signal. C14 blocks the DC content, R35 attenuates the signal and R34/C13 is a simple low-pass filter.

PCB FOIL PATTERNS (VERIFIED CORRECT)



Click on the picture to download the latest files.
Note: If that link fails, you can also get it from my SkyDrive here.

PDF Contents:
1. Silk layout
2. Keyboard layout

3. Top tracks
4. Bottom tracks
5. Top tracks - mirror image for acetate printing
6. Bottom tracks - mirror image for acetate printing

Hole coordinates are in the text file within the zip.

These are exact (as far as is possible) replica track and silk layouts of the original issue 2 board. Nearly all tracks and component placements on the board are within 0.5mm of the original. The majority of the tracks were laid out by taking high resolution scans of my original board. I had removed the modulator, heatsink and the ICs which were in sockets to ensure that I could see the hidden tracks. Tracks that were underneath the soldered-in ICs were visible if I viewed closely from the side of the chip, and again, I used a continuity tester to confirm. The layout of these "hidden" tracks was then updated to match a picture I obtained from a magazine article. Recently, I was provided a picture of a ZX80 with the keyboard layer removed (thanks Rich). With this, I was able to reproduce an accurate representation of the contact spirals underneath the keys.

A ZIP containing the PDF of the foil patterns and hole coordinates can be download HERE (currently version 1.8)
For list of changes to recent versions, please see the change-log at the end of this page.

Hole location coordinates and diameters for the foil is a plain text file. The coordinate origin is the bottom left of the board.
Format of the entries are X,Y,DIAMETER. Extract and reformat as necessary.

Martin has actually built a ZX80 using these foils (with the fixes mentioned above, that I had put into version 1.1) and it works !
Please see Martin's page at http://www.8bity.cz/?page_id=212

I have ALSO built a fully working ZX80 from the latest version (1.7) of the files (see below).

These are 600DPI prints in a PDF file (far higher resolution than you would ever need for etching), so you will need Adobe reader 7.0 or higher to view them. When printing from Adobe Acrobat reader, make sure you set Page Scaling to be "None".
I have included marks to indicate the board corners and I have also included rulers along the edges marked in millimetres to allow you to confirm the printout is to the correct scale.

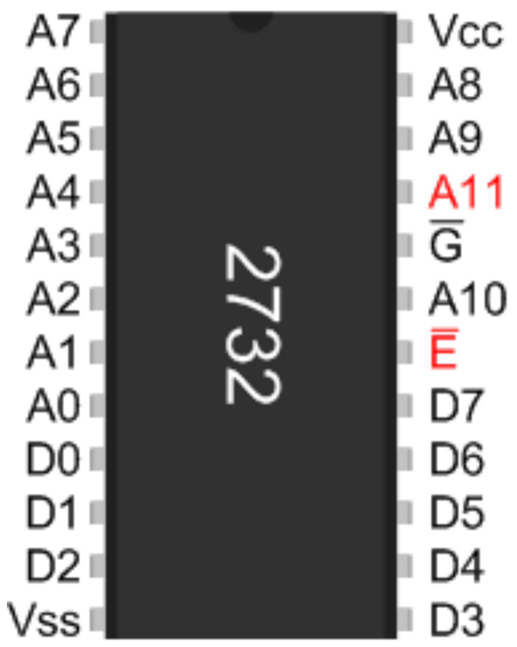
COMPONENT PLACEMENT

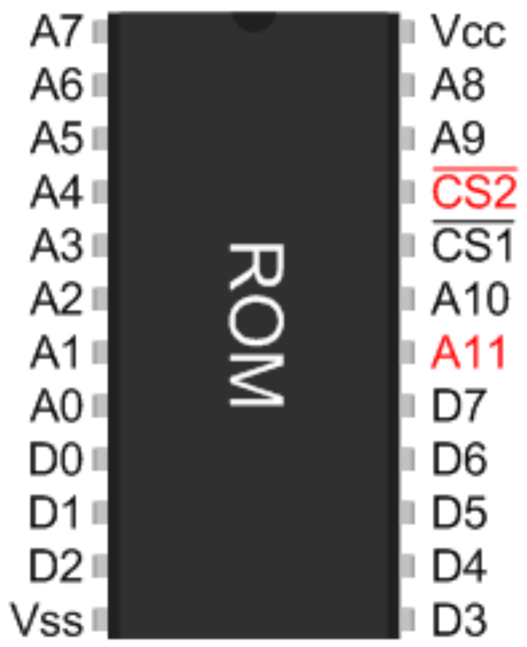
For component placement, I recommend that you refer to the original ZX80 below. Click on the image to enlarge it:

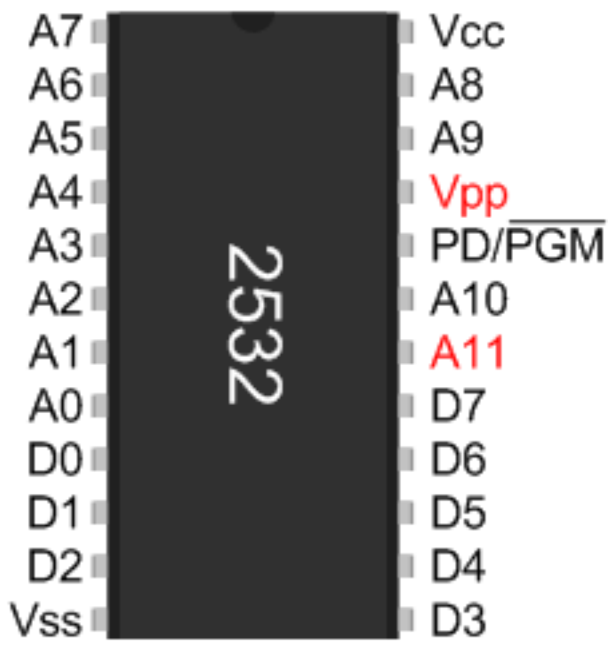
USING ALTERNATIVE COMPONENTS

Please note that my original board uses 2 x 2114 SRAM chips and a 2532 EPROM (which is slightly different pin-out to a 2732). So, if you cannot obtain them then you could either make an adapter board to plug into the original sockets, modify the PCB layout as appropriate or it may be possible to leave the RAM area of the PCB empty and build an add-on board to plug into the expansion bus instead.

The pin-outs for the 2732, the mask programmed ROM and 2532 chips here:

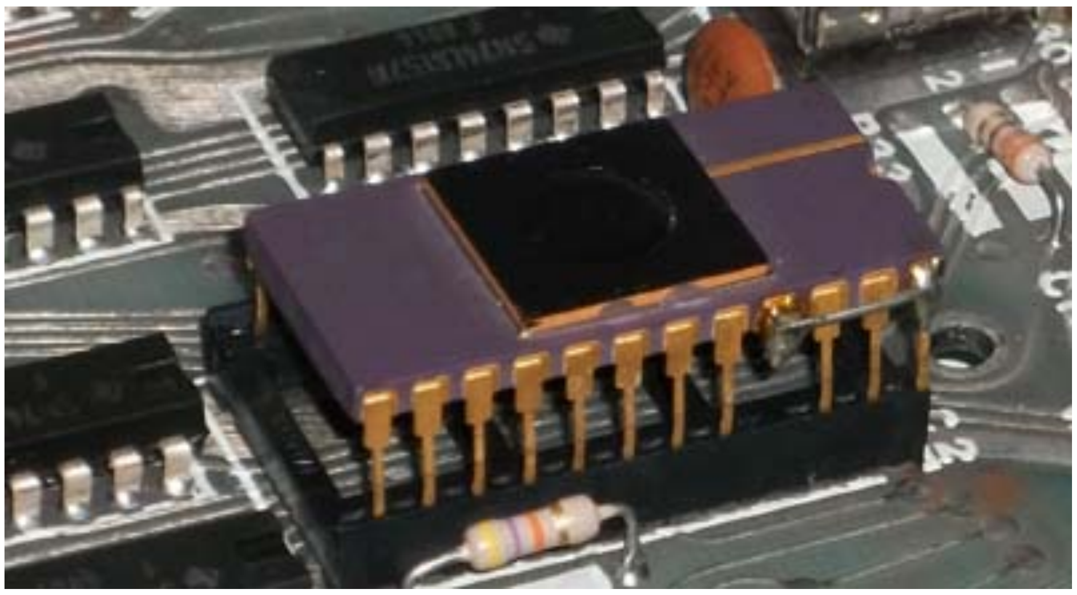






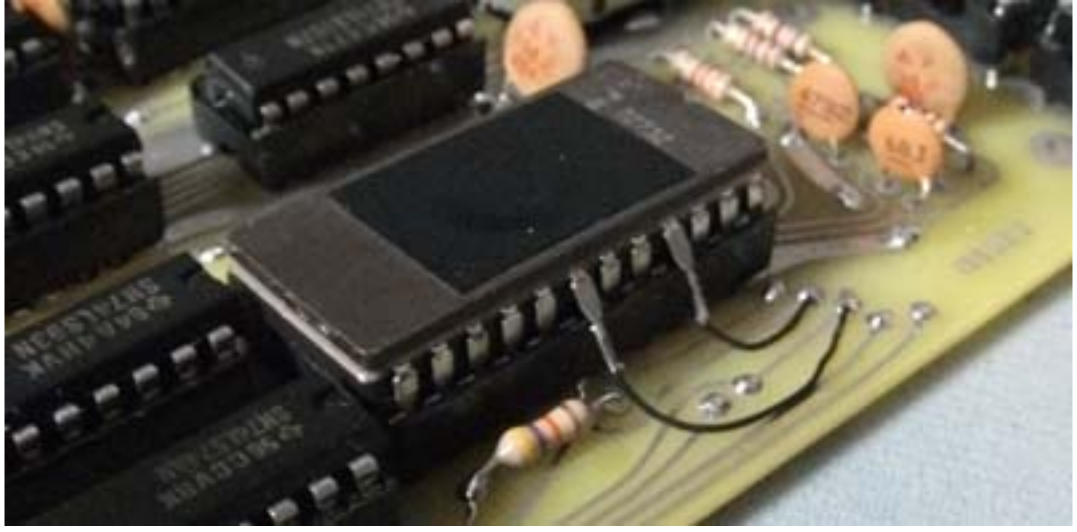
* - "/CS2" connection is the second chip select on the ROM version (normally connected to A12 so that the ROM is only enabled in the lower-4K of the 8K map).

My original ZX80 has a 2532 which does not have this second chip select so this pin on the 2532 EPROM is actually snipped and connected directly to Vcc, as shown here:

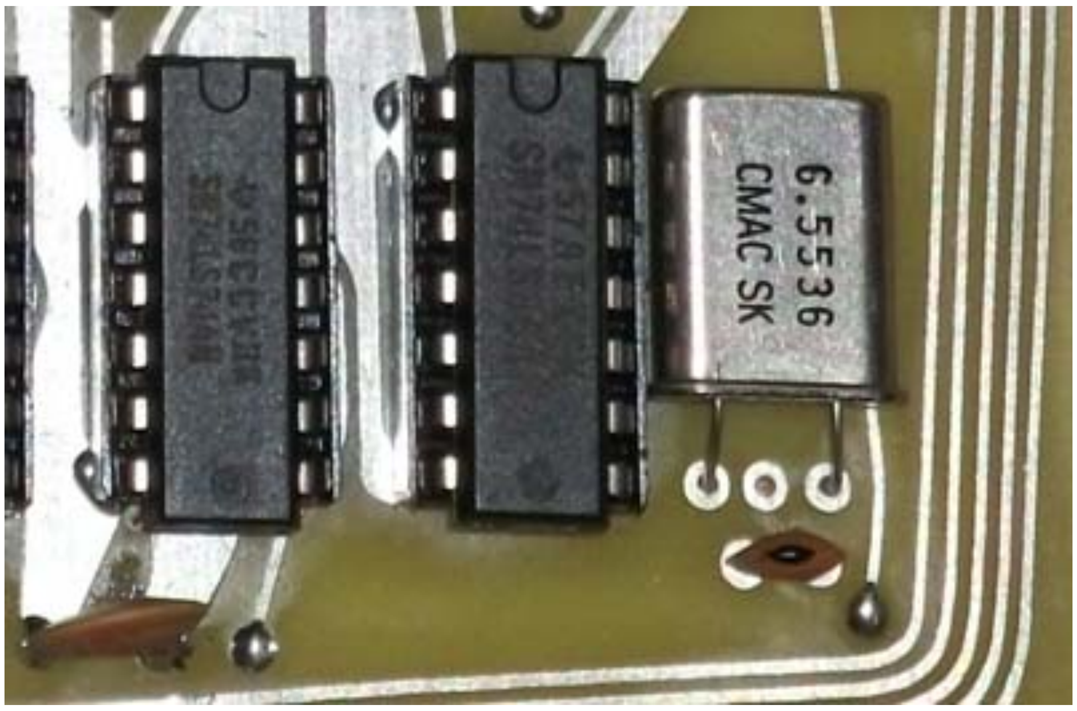


The 2732 EPROM does, however, have 2 chip selects (/G and /E) so both can be used with a small wiring change.
As you can see, there is a small difference on the right-hand side for the 2732. If you use a 2732 then you can either modify the layout or bend the 2 pins out and run short wires to the correct points (ie. swap them).

Here is how I did the modification to use a 2732 EPROM in my re-build machine. The A11 and /E pins are bent out and connected via short lengths of wire-wrap cable to the PCB that supplies the two signals...



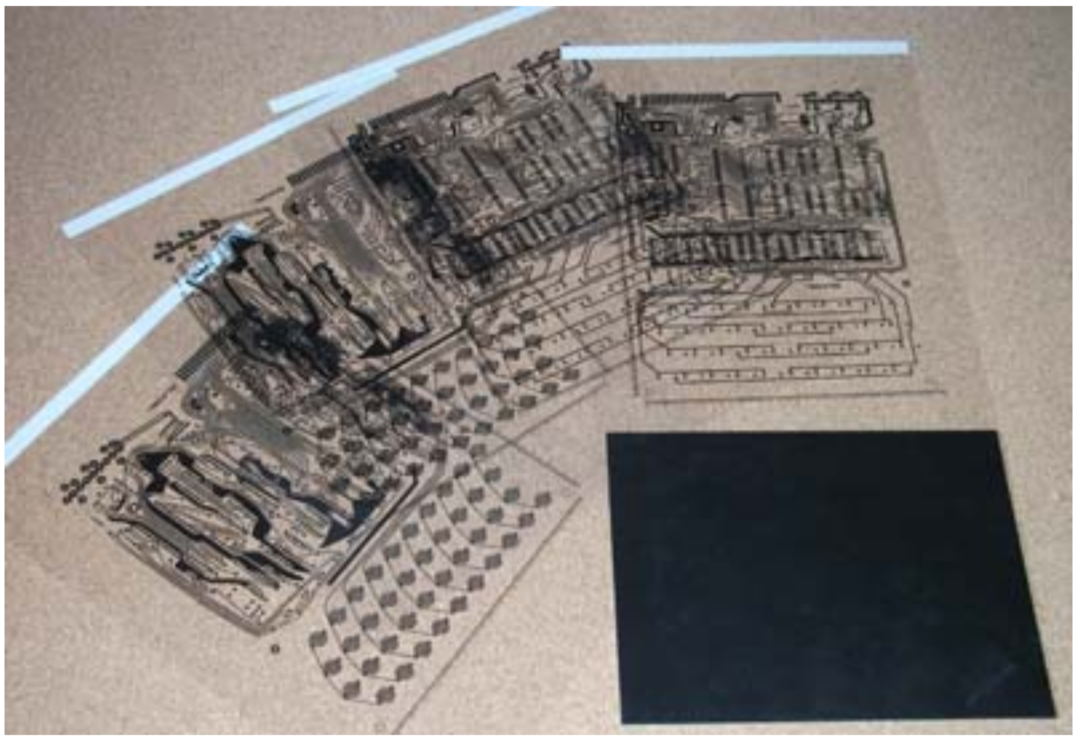
A crystal can be used instead of a ceramic resonator, and it fits very nicely on the board (shown resting on top of R27 to the right of IC20):



CONSTRUCTION (ETCHED PCB VERSION)

Cut the board to the exact size required, in this case, 204mm by 155.5 mm
The PDF files contain the required artwork that needs to be printed. This contains both normal and reversed images. Make sure you print the correct ones as needed.

Print the track images onto acetate sheets. If using a laser printer, then normally two copies, stacked, are sufficiently dark enough to block the UV light in the exposure box. These should be reverse-printed images, ideally, if using acetates as this will allow the ink-side to be in direct contact with the board and will ensure a sharp image being transferred.



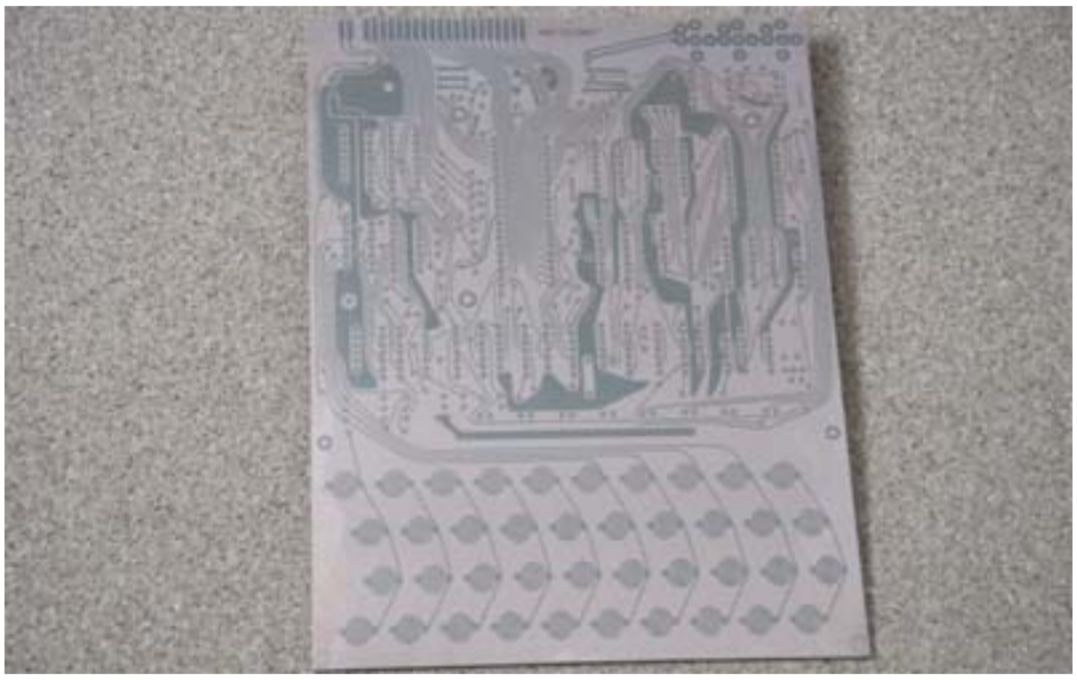
Lay ALL of the acetates on top of one another to make sure they are line up. My laser printer slips a little bit, so I had to print a few extra until I managed to get a set where all were aligned with each other. The corner index markings allow perfect alignment.

Check the sizing by overlaying the acetate on the cut PCB...

This was exposed in my home-made UV exposure unit, with two 8W UV tubes.
The exposure time was only 75 seconds for each side.
Test with an off-cut with different timings and pick one which cleanly exposes the board. Do not over-expose as the thin tracks will disappear.
When both sides have been exposed, develop the board immediately. I recommend using proper PCB developer, as the developer timing is much more relaxed than using the home-made mixes.

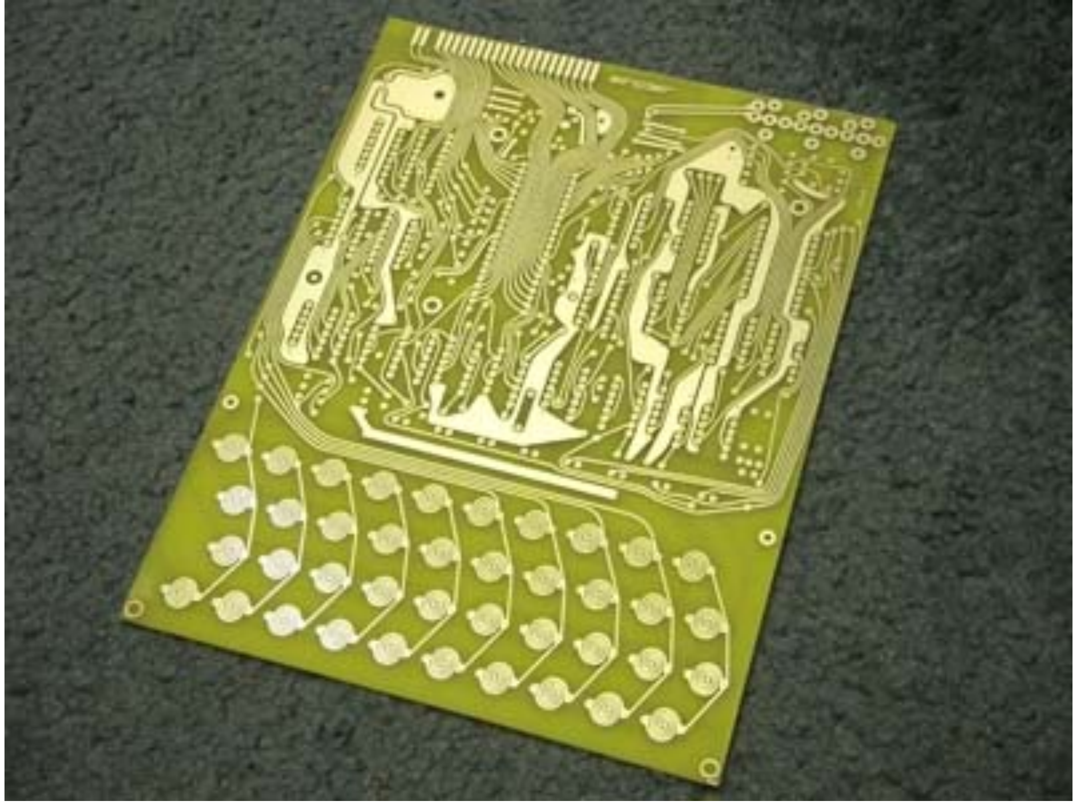
This artwork contains index marks at all corners - you will need to accurately align the board to these to ensure the top and bottom holes are on top of each other. The tracks are quite tight so accuracy is essential.

Once exposed and developed, the board will look as follows. The parts to be etched will be bare copper, and the resist remains where the tracks are to be...



Etching of the board took approx 25 mins with a fresh warm solution of ferric chloride. Parts of the board were etched quite some time before this, but the centre of the more exposed areas always seem to take much longer. The board was removed and rinsed as soon as the areas were clear. Any longer would cause the thinner of the tracks to be eaten into.

Once etched, it will look like this...

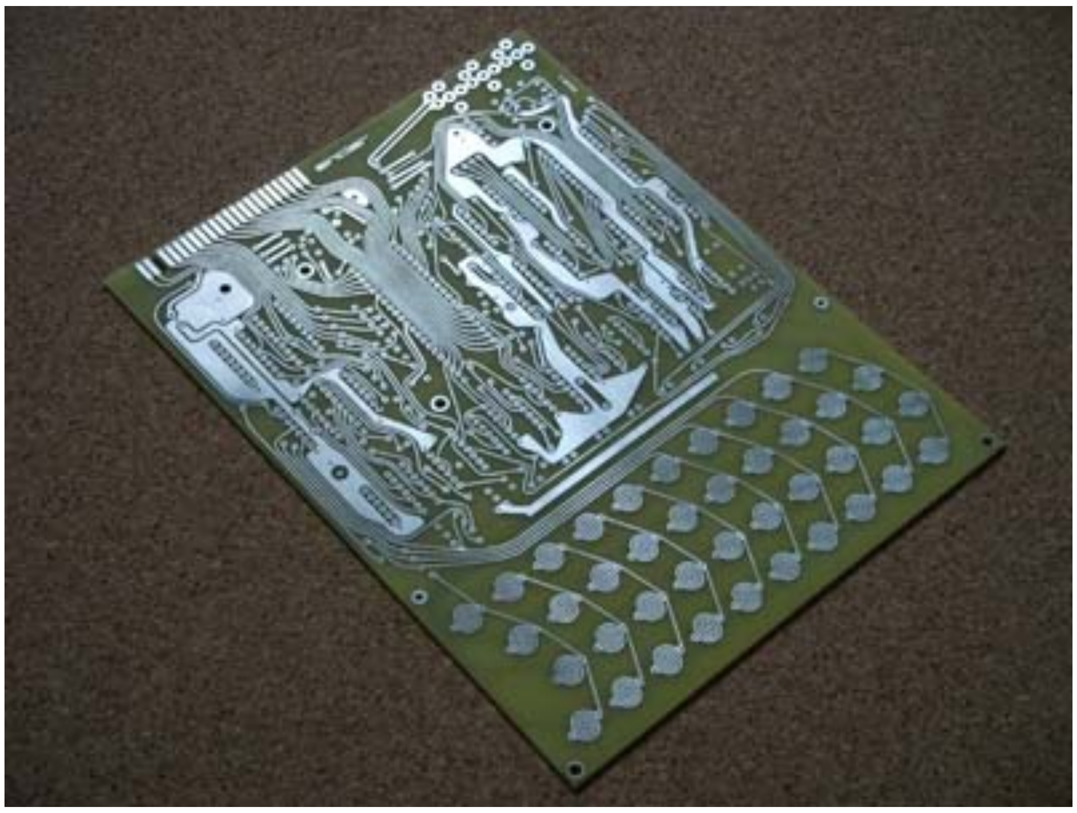


With careful etching, very fine-detail results can be achieved, as shown here with the Sinclair lettering (the "s" is only 3.25 mm wide and the dots above the "i"s are only 0.4mm)...

The complete board was then drilled.
A 0.8mm drill is used for the majority of the holes. A stand is really recommended for the drill as it will then be held vertical. Using a handheld drill is likely to result in the holes in the back of the board being off-vertical, so will not be positioned at the centre of the rear pads.

Check the board very carefully with a magnifying glass to see if there are any bridges between tracks or pads. These are removed at this stage using a sharp knife. Spending some time at this stage will result in a machine that should work first-time.

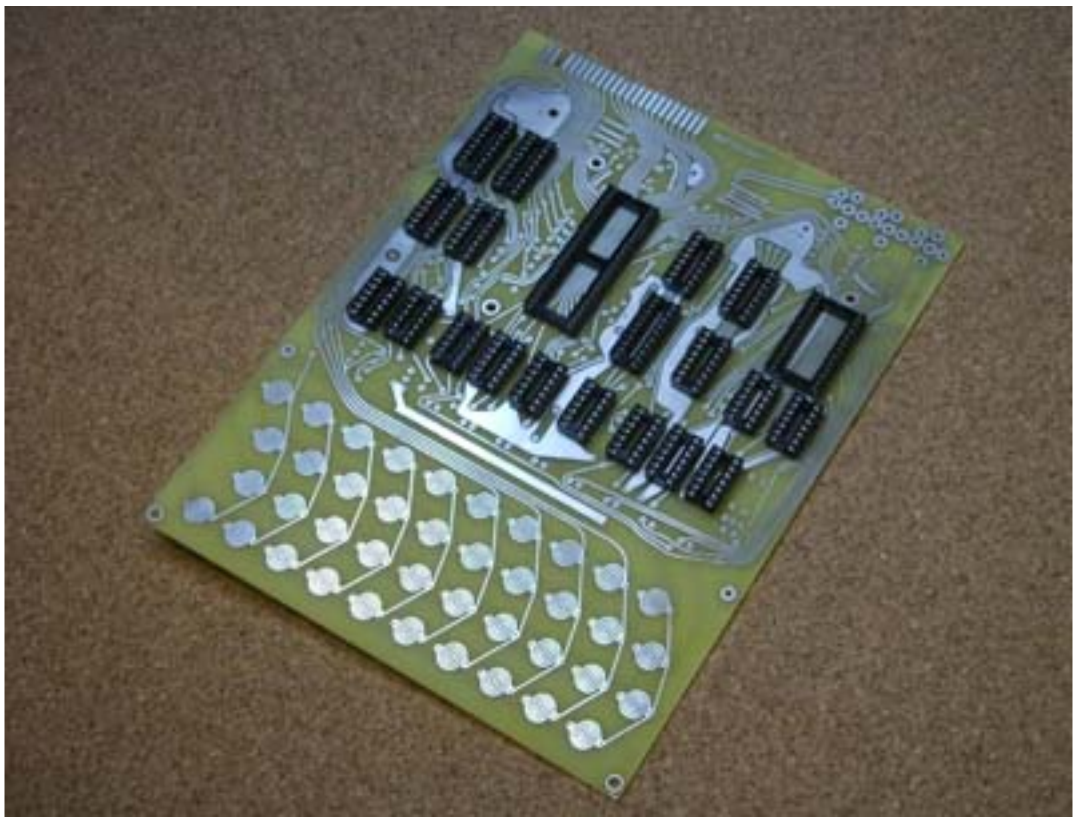
As this is to be left open, I chemically-tinned the complete board. The chemicals are not very cheap, but you don't need to mix the whole quantity at once, and the unmixed crystals have a very long shelf-life. The tinning was well worth it, as it gives a professional tinned finish to the board that will keep it looking bright. It also ensures the keypads won't corrode and give poor contact.
Hope you agree the results are worth it...



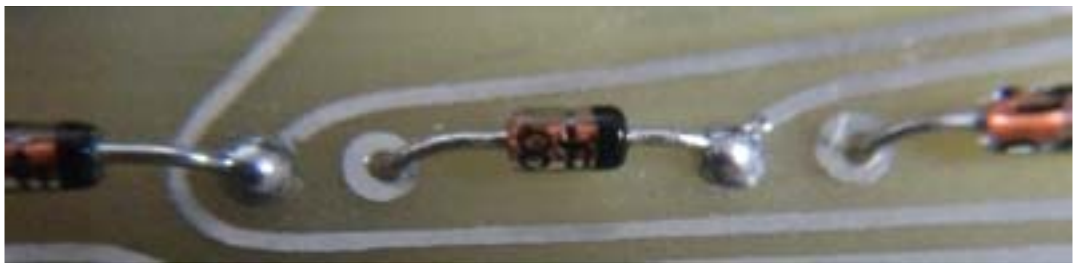
Sockets need to be soldered. This was definitely the most difficult and time-consuming part.
I wanted to use standard sockets (as I had them spare) and I didn't have plated-through holes, so I needed a way of ensuring that the pins that required connections to the top and bottom tracks were connected.
To do this, I did the following...
1. Take a strand of stripped wire-wrap wire and push it upward from the back to the front of the PCB.
2. With about 5mm protruding from the front, I gently pushed a wooden cocktail stick into the hole to trap the wire. This also serves to stop the solder flowing into the hole.
3. Using a small screwdriver, I pushed the wire around the stick to form a small loop, touching the board.
4. This loop, with the cocktail stick still in place, was soldered to the top layer pad.
5. The stick is then removed to reveal the hole.
6. Cut off excess on the back (don't solder) leaving about 5 mm.
7. Repeat this for the other holes in the socket that need soldering to the top layer.
8. Once all required holes done, the chip socket is pushed into the holes. There is sufficient space for the socket legs and the wire to pass through the same hole.
9. With the chip socket firmly pressed against the top of the board, the bottom pads are soldered as normal, soldering the wire and the socket leg together.

Alternate ways of soldering sockets onto a home-made double-sided board:
1. Purchase wire-wrap sockets. These have very long pins so you will be able to mount the sockets a small distance from the board, allowing sufficient space to enable a fine-tipped soldering iron to complete the top layer solder joint.
or
2. (Cheaper than wire-wrap option) Buy turned-pin sockets then use a pair of pliers to squeeze the plastic on the socket, breaking it and allowing the metal pins to be removed. Then, take a spare chip (or another socket), push these metal sockets that you have extracted onto the pins. You can then insert the chip/socket into the PCB as normal and solder the top and bottom track connections to the individual sockets. Then, pull out the chip and the soldered socket pins will then be neatly aligned and soldered. Konstantinos has built a ZX80 with an earlier version of the PCB foils presented on this page, and has added some very clear pictures of the procedure that I have described on his website here.

Also, all vias that are underneath the chips are soldered before the socket is attached, as these can't be accessed later.

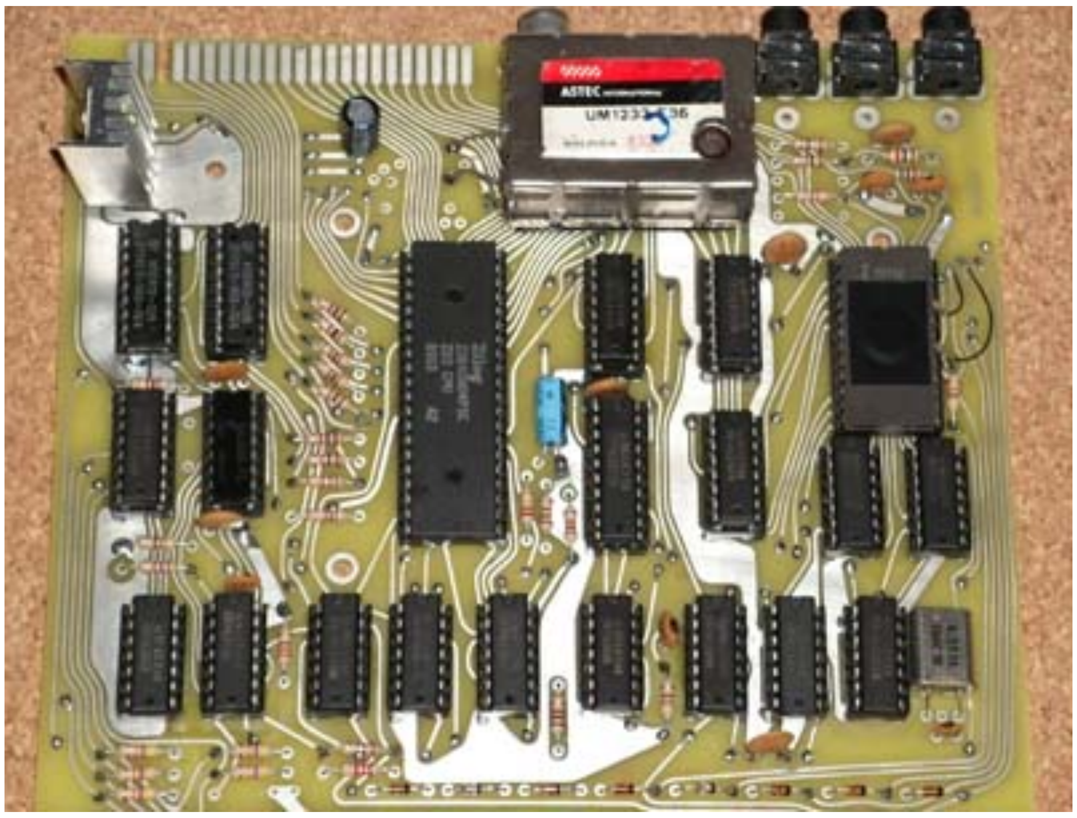


Then the remainder of the components were then soldered into place. Some only need soldering to the bottom tracks. Where a track is also present at the top, the component is soldered to this as well, as shown here...

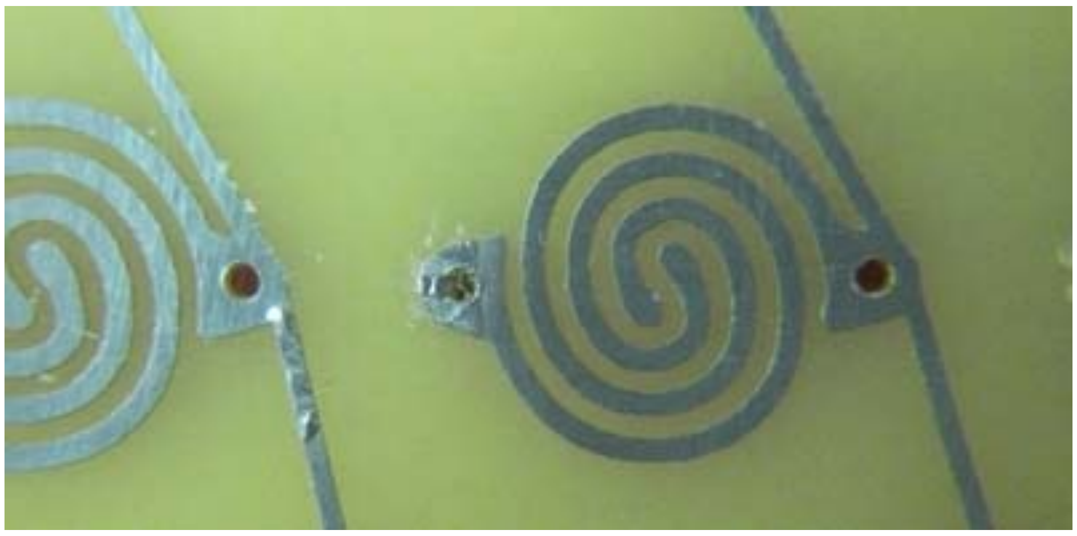


Some components (the capacitors) need to be a couple of mm off the board to allow this.

Once all components are fitted, all remaining holes that have tracks attached to them both above and below the board must be joined. In this picture a temporary heatsink is fitted to the regulator until I got some aluminium.



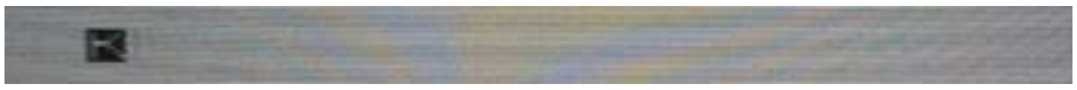
I wanted the keyboard to lay flat. If the holes were through-hole plated then this would not be a problem. I, however, needed to solder the top layer through to the track underneath while still ensuring a flat keyboard could be used. To do this, I used a strand of wire-wrap then squeezed it flat using a pair of pliers. This flattend wire could then be soldered using a minimal amount of solder and some very fine soldering. The excess was then removed using desoldering wick. The result is shown here. If you look carefully then you can just see the flat wire soldered to the edge of the centre hole in the picture.



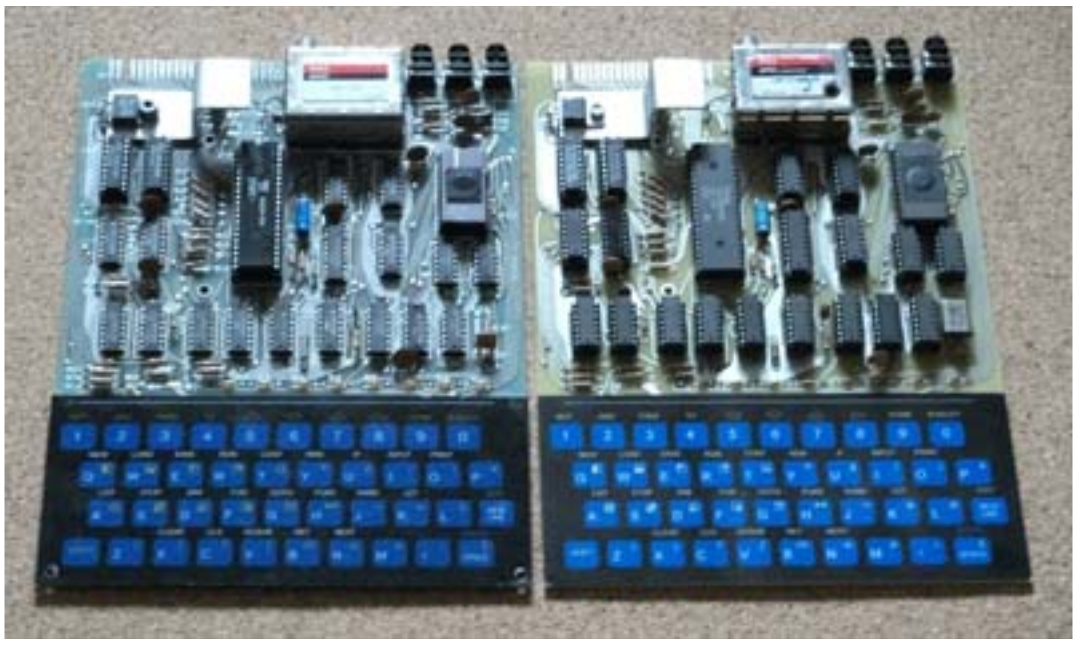
The keyboard was printed onto photo paper then laminated to make it durable. It was then trimmed and the contacts stuck on the back. Once the contacts were attached, a spacer sheet of acetate was attached with holes to allow the foil to make contact with the PCB contacts when pressed. Everything is fixed using double-sided tape.

Once all components are soldered and without any ICs inserted into the sockets, connect a continuity tester between the power supply pins. If all is well connect a power supply (7.5V to 12V, current regulated to 500mA if possible). Check the +5V and 0V connections on each IC socket (normally top-right and bottom-left respectively, except for IC21). Turn off the supply and insert the ICs. Turn on the power supply and connect to a television on Channel 36 (approx) if using the normal modulator. If the circuit is working then expect a power consumption of around 350mA.

If using a TV then adjust the tuning to get a picture (note: the display of black text on white **will require the brightness control of the TV to be turned up**). A working circuit should show an inverse "K" in the bottom left of the screen.



Here is my completed (fully working) rebuild next to the original...



That's it. A fully working replica of the original.

ROM IMAGES

The ZX80 ROM is 4K long.

Click here to download the ROM in binary format

You can also use the ZX81 ROM image (8K):

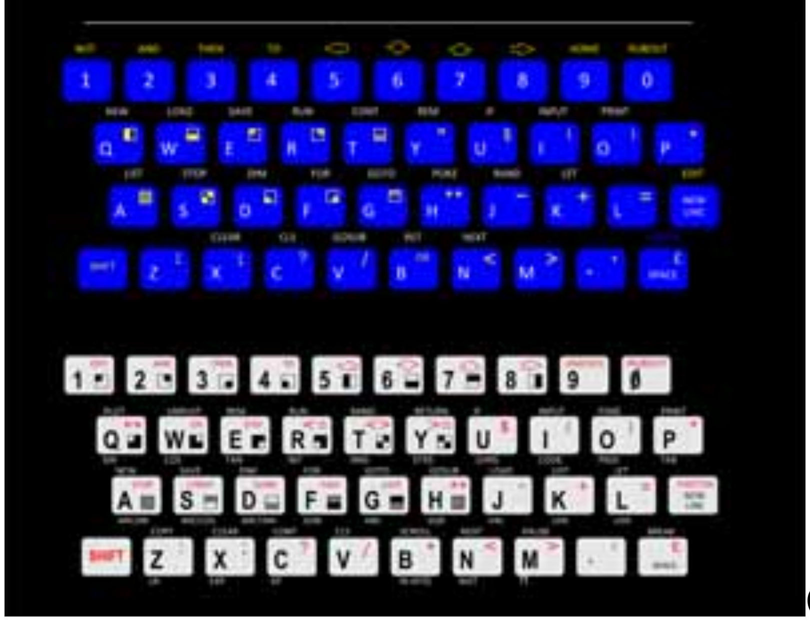
Click here to download the ROM in binary format

For the ZX81 ROM to work you will need to use a 2764 EPROM (28 pin instead of the original 24 pin, so will need to adapt as appropriate) and must take A12 to the appropriate address line on the ROM and ground the /OE pin.

On my stripboard version I actually now use a 16KB ROM with BOTH images blown into a 27128 EPROM with a switch on the A13 line to determine whether to run the ZX81 or ZX80 image.

KEYBOARD OVERLAYS

Accurate keyboard overlays (same size as original, suitable for use with the PCB foils shown above) are available here.


Click on the image to view a 600DPI version.

I have spent a lot of time on these overlays to make them as faithful to the originals as possible.

When you click on the image using Internet Explorer or Firefox, the large version will be displayed scaled to fit the window, and will not show them at their best. You normally need to click on the image which is then displayed to zoom in full-size.

Higher resolutions are available (as I drew them using vector graphics so I can rescale as much as required). Please let me know if you want them.

For best results, these need to be printed onto glossy photo paper.

CONSTRUCTION (STRIPBOARD VERSION)

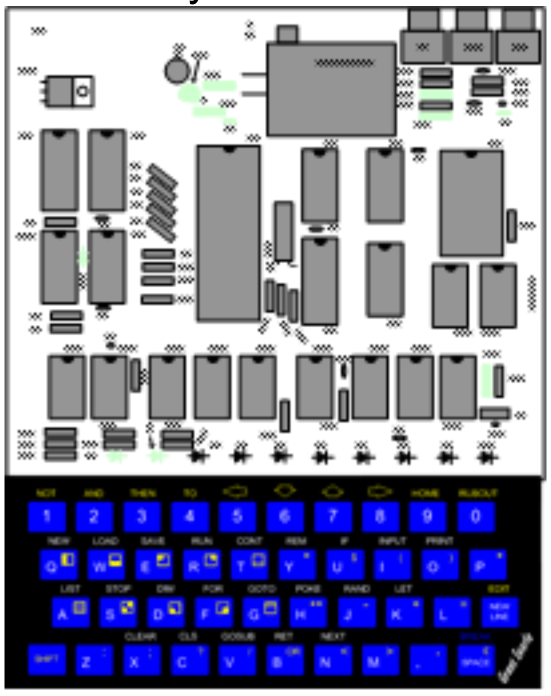
The complete computer is built on a large piece of perforated stripboard. All digital connections are made using wire-wrap wire as this is a lot thinner than conventional hook-up wire so the data and address bundles can be tied to form a neat layout. The positioning of the ICs in relation to each other matches the positioning in the original ZX80.

I recommend printing the circuit out on an A3 sheet if possible. The pin numbers are easier to read on my re-drawn schematic than on the original scan, and my version has corrected the errors found on the original version.

Begin construction by positioning the IC sockets (see my pictures below for guidance), ensuring there is sufficient space between then to make all of the connections, and position the remaining components.

I would recommend that you adopt a layout similar to Sinclair's (that is what I did).

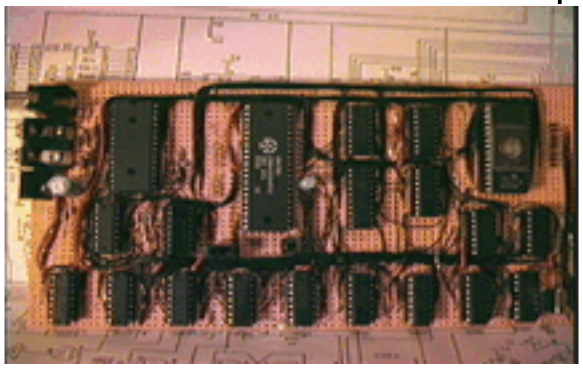
Here is my re-drawn version of the layout found in the ZX80 assembly manual:


Click on this image to see a large version.

When you click on the image using Internet Explorer or Firefox, the large version will be displayed bit it will be scaled to fit the window, and will not display the image at it's best. You normally need to click on the image which is then displayed to zoom in full-size.

The pale-green components are not actually used in the circuit, but there are spaces for them on the PCB.

Here is the ZX80 build on stripboard:



Click on this image to see a large version.

As you can see, the component placement is in a similar manner to the original. The two original RAM chips replaced with a single larger one instead.

Next I recommend wiring the data and address buses for the top half of the circuit. As each wire is soldered it, mark it off on the circuit diagram. This helps avoid any errors resulting from missing connections. Wire in the logic ICs, resistors, capacitors and diodes. As virtually all pins on every IC are to be connected, any errors made will become obvious as more connections are made. For initial testing, there is no need to wire the keyboard into the circuit.

Once all components are soldered and without any ICs inserted into the sockets, connect a continuity tester between the power supply pins. If a short exists then check the underside of the board for any missing track breaks or flakes of copper between tracks. If all is well connect a power supply (current regulated to 500mA if possible). Check the +5V and 0V connections on each IC socket. Turn off the supply and insert the ICs. Turn on the power supply and the television or monitor. If the circuit is working then expect a power consumption of around 310mA. If using a TV then adjust the tuning to get a picture (note: the display of black text on white will require the brightness control of the TV to be turned up). A working circuit should show an inverse "K" in the bottom left of the screen. If all is well then wire in the keyboard if not already done.

PARTS LIST

This is a full list of discrete components required to build the ZX80 excluding any hardware such as power supplies, nuts and bolts etc.

Component No	**Type**	**Qty**	
IC1	Z80A	1	
IC2	2732 4K EPROM	1	(Or larger with modification)
IC3/4	2114 1Kx4 bit RAM	2	(Or can use 62256 with modification)
IC5	74LS373	1	
IC6-8	74LS157	3	
IC9	74LS165	1	
IC10	74LS365	1	
IC11,12	74LS00	2	
IC13	74LS04	1	
IC14,15	74LS05	2	
IC16	74LS10	1	
IC17	74LS32	1	
IC18,19	74LS74	2	
IC20	74LS86	1	
IC21	74LS93	1	

IC22	7805	1
D3-10	1N4148	8
X1	6.5MHz Xtal or ceramic resonator	1
C1-6,12,13	47nF	8
C8	22uF	1
C9,11,14, 15	47pF	4
C10	1uF	1
R29	180R	1
R35	1M	1
R1,3-11,18, 22,23,27, 30,34	1K	16
R32	330R	1
R21	220K	1
R13-16,?	47K	6
R2,25	470R	2
R19	2k2	1
3.5mm Jack Socket		3

Miscellaneous, depending on build

SW1-40	SPST Push	40
Heatsink for regulator		1
Wire wrap cable for interconenctions (Stripboard)		1 reel (25m)
Vero Board/PCB		1
40 Pin DIL socket		1
20 Pin DIL socket		1
16 Pin DIL socket		5
14 Pin DIL socket		11
18 Pin DIL socket (2114 RAM)		2
28 Pin DIL socket (62256 RAM)		1

PICTURES OF FINISHED STRIPBOARD VERSION

Full component side view
Close-up of component area
Close-up of the keyboard
Reverse of circuit board
The connectors used

SOFTWARE

ZX80
Follow this link to go to my ZX80 software page, featuring a flicker-free space invaders game for you to type in. Search the net to find some other flicker-free games for the ZX80.

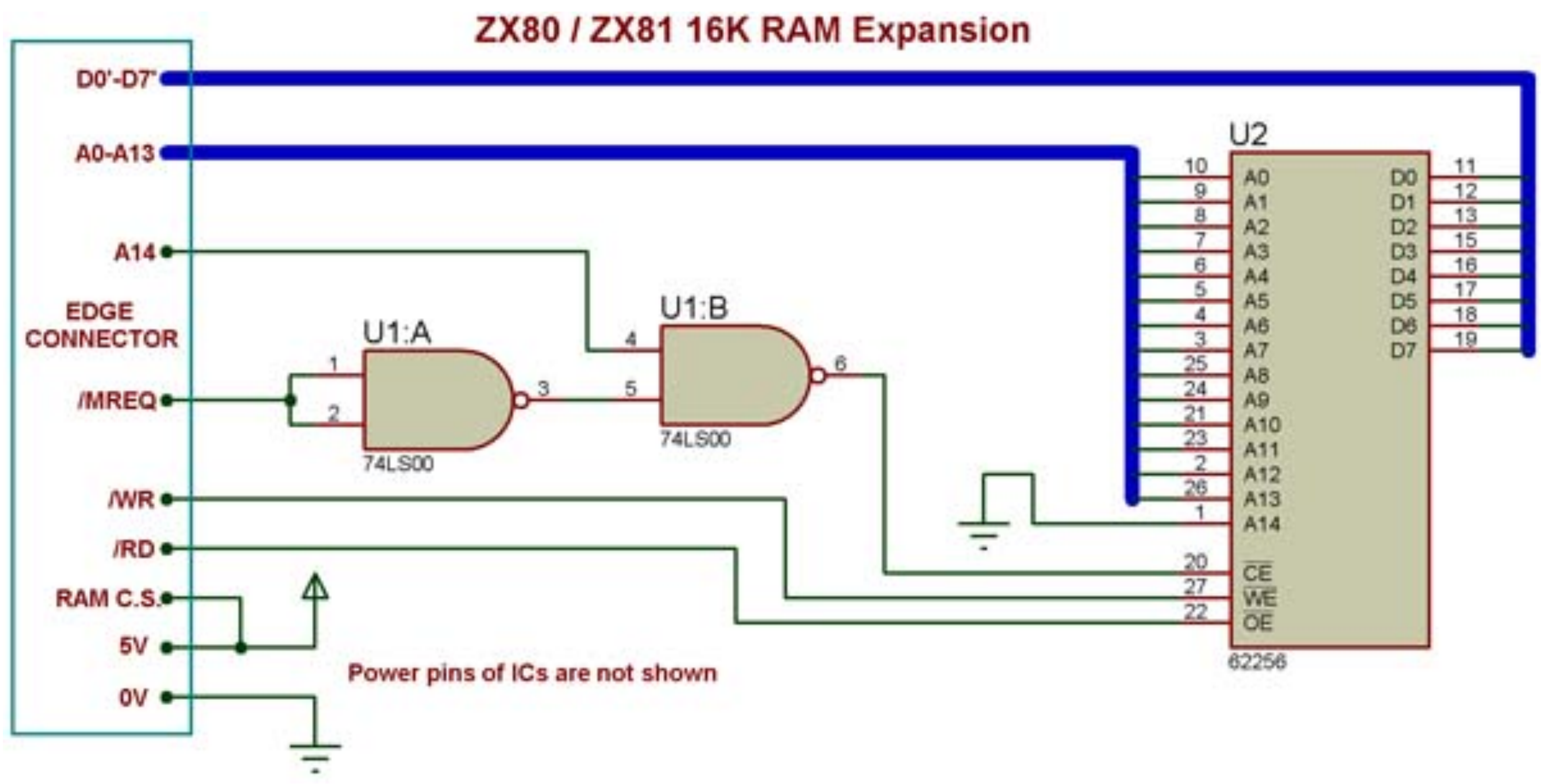
ZX81
A software archive for the ZX81 is available at ftp.nvg.unit.no. These files are in the ".P" format as used by XTENDER, the ZX81 emulator. To use these on a real ZX81, you can build a VERY simple

cable connected to the parallel port and use this program by Wilf Rigter to send the files to the ZX81.

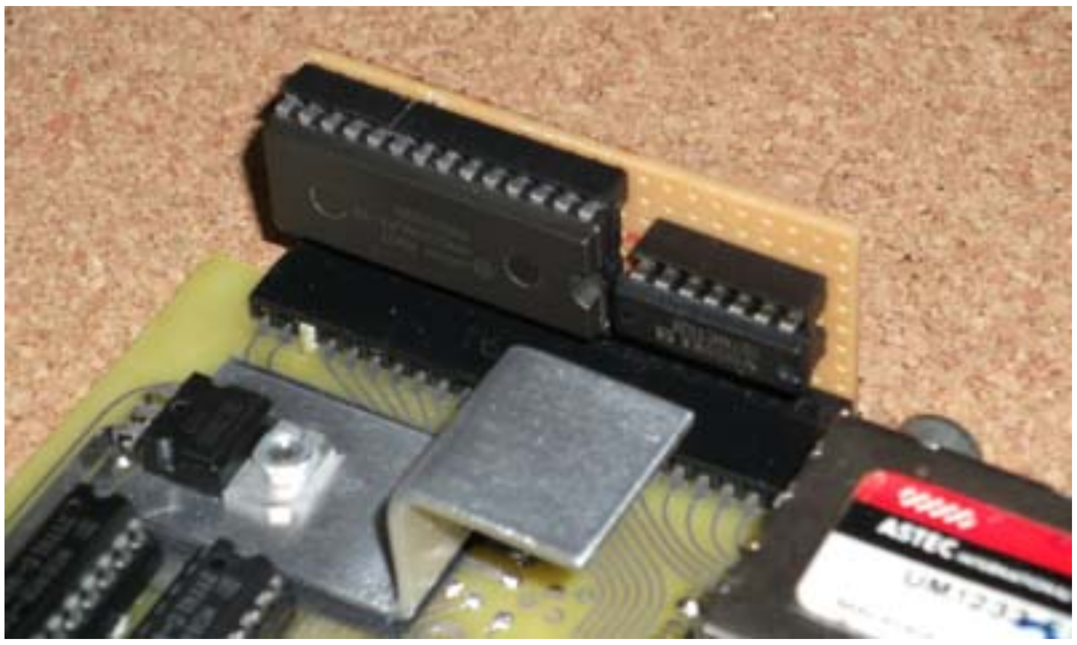
SIMPLE UPGRADES

16K RAM EXPANSION

This can be built with just two ICs - 74LS00 (or a 74LS02 with a circuit modification) and a 62256.


ZX80 / ZX81 16K RAM Expansion
D0'-D7'
A0-A13
A14
EDGE CONNECTOR
/MREQ
/WR
/RD
RAM C.S.
5V
0V
U1:A
74LS00
U1:B
74LS00
U2
62256
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
D0 D1 D2 D3 D4 D5 D6 D7
CE
WE
OE
Power pins of ICs are not shown

The prototype shown here is a very compact layout which is not much higher than the heatsink.

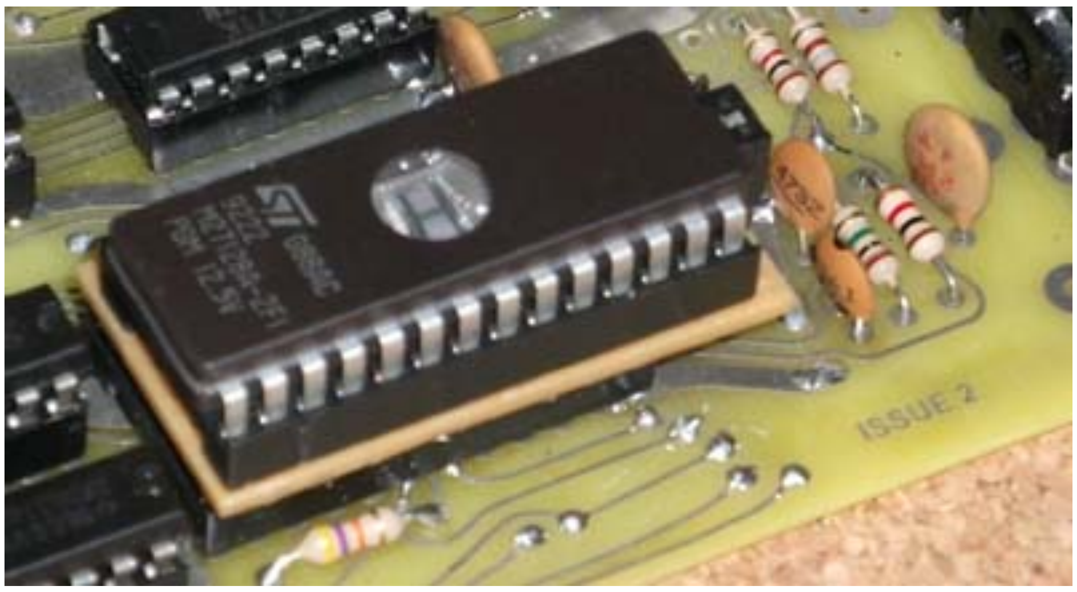


This was built on "tri-pad" prototyping board, with "wire runner" connections on the back.

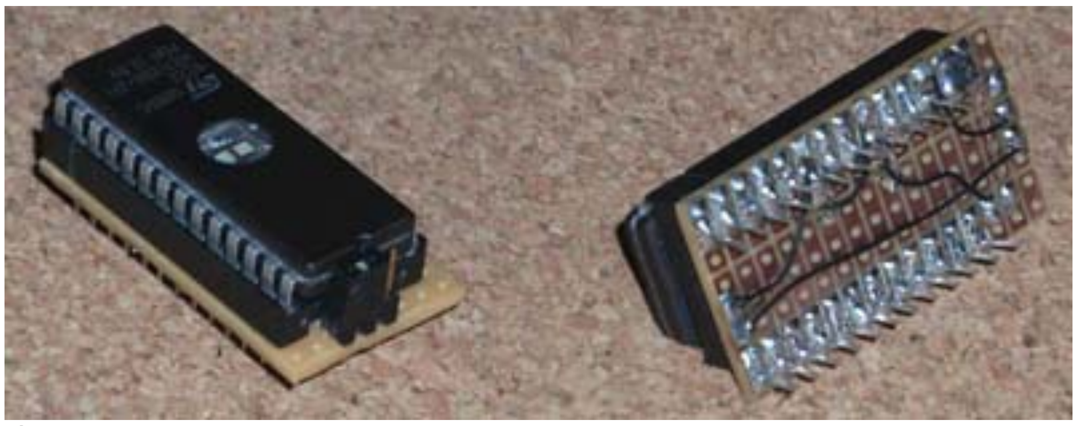
Here is a view of the front and back...

ZX80/ZX81 27128 EPROM fitting

The 27128 is a 28 pin device, but the board only allows for a 24 pin ROM, so a piggy-back board is required to allow a larger device to be fitted.



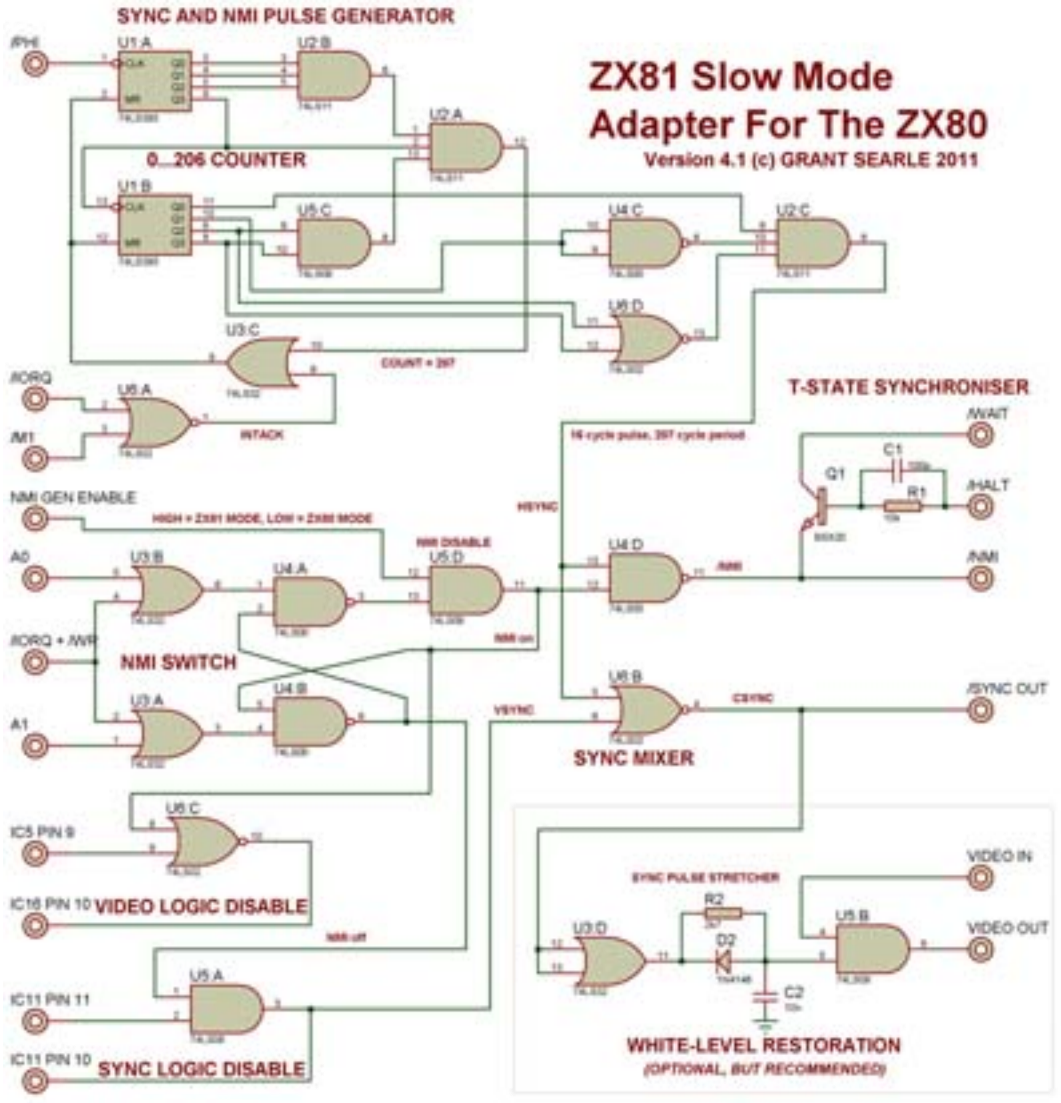
There is a jumper on the board to allow the top address line (A13) to be switched between 0V and Vcc. This way, both the ZX80 and ZX81 images can be on the same EPROM (I recomment ZX80 ROM occupying 0K-4K, ZX81 ROM occupying 8K-16K), and the jumper allows selection between the two. If you are building the ZX81 NMI generator (ZX80 to ZX81 upgrade), then this A13 address line can be connected to the NMI GEN ENABLE input so that the NMI generator is automatically enabled if the ZX81 ROM is enabled. Below is a view of the top and bottom of the adapter board. Many pins are in the same positions as the original ROM. The others are wired to the correct pins on the EPROM.



Connections:
No circuit diagram for this, as it is a straight transfer of connections, as shown below:
EPROM A0 to A11 --> ROM Socket A0 to A11
EPROM A12 --> ROM Socket /CS2 (which is actually A12)
EPROM Vcc, Vpp and /P --> ROM Socket Vcc
EPROM Gnd and /E --> ROM Socket Gnd
EPROM /G --> ROM Socket /CS1
EPROM A13 --> Jumper to connect to either Vcc or Gnd

ZX81 CONVERSION

Six ICs, a small number of other components and a ZX81 8K ROM is all that is needed to convert the ZX80 into a fully functional ZX81 which supports both SLOW and FAST modes using the same timings and synchronisation as a real ZX81.



Click on image to go to page

Details here to see a full description of a circuit to convert the ZX80 into a fully functional ZX81 (including SLOW mode).

THE ORIGINAL ZX80 KIT

Some pictures of the ZX80 in kit form that I have seen in my magazines.

Picture from "ZX Computing" (Issue 1) - (Carefully re-touched, as it was over 2 pages in the magazine) - very big!
Picture of kit

LINKS

My pages

Build your own ZX80 - my page showing you how to build this old micro
  |__ ZX80 to ZX81 conversion - build the NMI generator needed to convert the ZX80 circuit into a ZX81
  |__ ZX80 software - Type in a Space Invaders game into the ZX80

Build your own Jupiter Ace - my page showing you how to build this old micro
Build your own UK101 - my page showing you how to build a greatly simplified version of this old micro
Pong - Pictures of my build of the Atari classic arcade game
My Machines - My collection of classic 80's micros

Other user's pages

ZX81 Rules - Wilf Rigter's excellently detailed pages for ZX81 circuitry. He was a great help with

the ZX80/81 video details
Rodney Knapp's homebuilt micros - lots of information on the ZX81 computer which he has built
The German ZX-TEAM - Sinclair enthusiasts page by Peter Liebert-Adelt
"The" software archive for the ZX81 - at ftp.nvg.unit.no

CHANGES

TRACK LAYOUTS
Changes for 1.1 *(Many thanks to Martin Lukasek for pointing out the errors):*
Link missing which meant the keyboard resistors were floating, intead of being connected to +5V
The link from the lower ICs was connected to IORQ instead of MREQ on the Z80
I have used an actual magazine picture (carefully rotated/stretched etc) of the bare PCB to allow me to reposition the hidden tracks under the ICs. No change to functionality, but it makes the layout more accurate to the original.
Corrections to the foil for 1.1 can he seen here
Changes for 1.2
Update to use round pads on the upper IC pins instead of oval to avoid touching nearby tracks, and some small graphic touch-ups.
Changes for 1.3
Several graphic touch-ups to increase accuracy. PCB resized slightly to 155.5mm x 204mm to accurately match the size of the original.
Images within the PDF use higher quality compression to reduce JPEG artifacts (bigger size PDF, but worth it).
The two mounting holes at the bottom of the keyboard changed to accurately match the original.
Changes for 1.4 *(Thanks to Rich at www.rwapsoftware.co.uk for providing a picture of the PCB underneath the keyboard. He also sells replacement ZX80 keyboard overlays)*
Keyboard pads changed to spirals (as you can see on the small image above) instead of a fingers to more closely match the contacts under the real ZX80 overlay.
Silkscreen layout.
Track images no longer JPEG compressed.
The "origin" of the rulers is now bottom-left for the top layers and bottom-right for the bottom layer, as it is viewed from underneath.
Changes for 1.5
Full review of the rear tracks - several updates to improve/correct spacing between tracks and pads. General tidy-up.
Changes for 1.6
Very minor (I'm being fussy!) alignment and spacing updates to ensure tracks between pads are in the middle and parallel track spacing is improved..
Changes for 1.7
PDF file now contains 2-colour images of the tracks to improve printing (to eliminate the anti-alias shading). Reversed images also included for acetate printing.
Changes for 1.8
Some of the front tracks under the keyboard overlay and under some of the chips have been moved slightly to accurately match a high-resolution scan of a bare board that was sent to me. Thanks to Andrew Gale - most appreciated!

I hope this page has been useful.

Grant.

To contact me, my current eMail address can be found here. Please note that this address may change to avoid spam.