**Building the N8VEM Single-Board Computer for the Total Beginner**

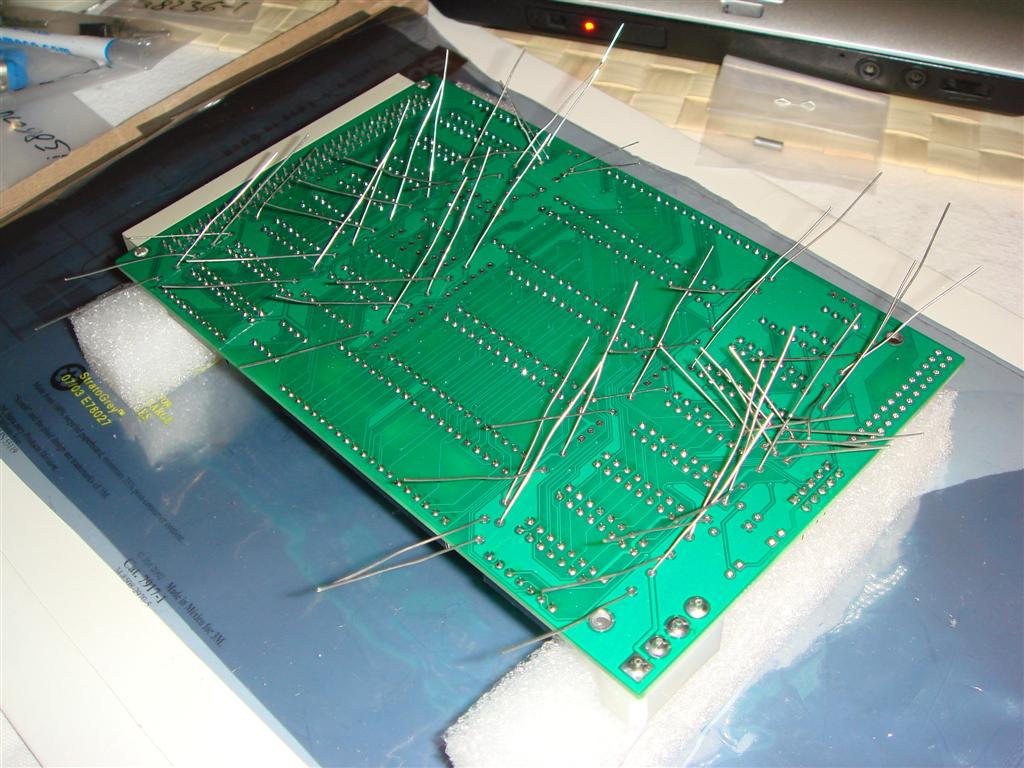
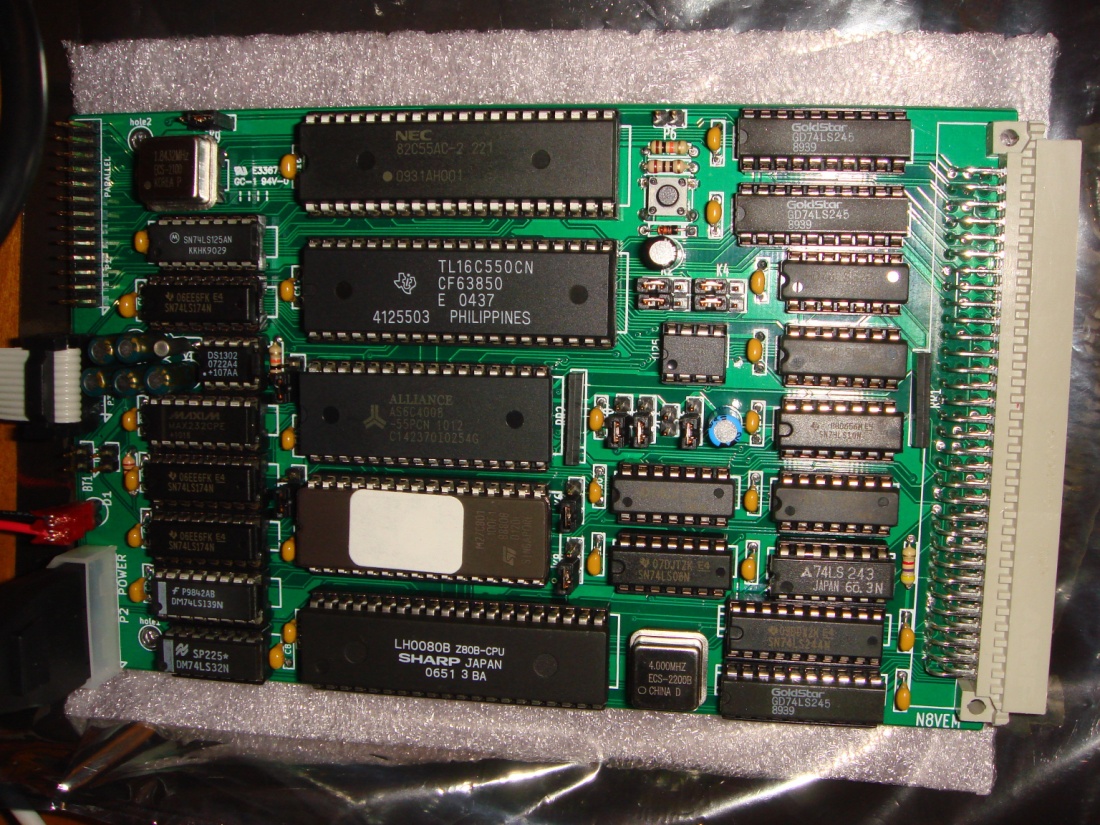
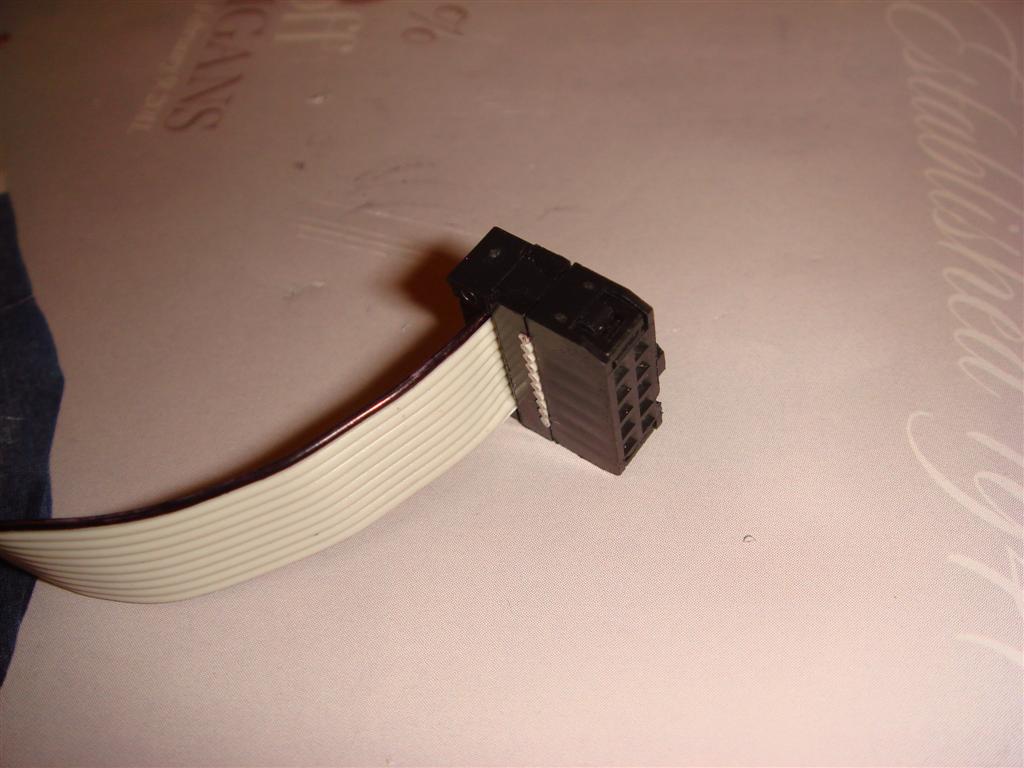
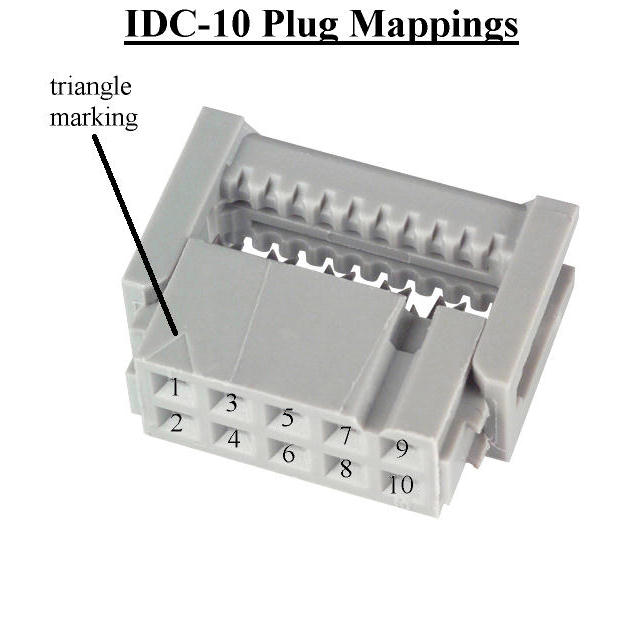
This is a guide to help with building the N8VEM Single-Board Computer (or SBC). It is aimed at the total beginner, and I will do my best to not assume much of any previous electronics experience or knowledge.

I built my 1st circuit board just a month or two before my N8VEM, and I had no electronics experience prior to that. I had a solid foundation in programming, but did not know much about hardware side, however, I was motivated to learn. I found the construction process to be a bit confusing at times, but it is my hope that this guide will help those who have limited experience with these things as I did.

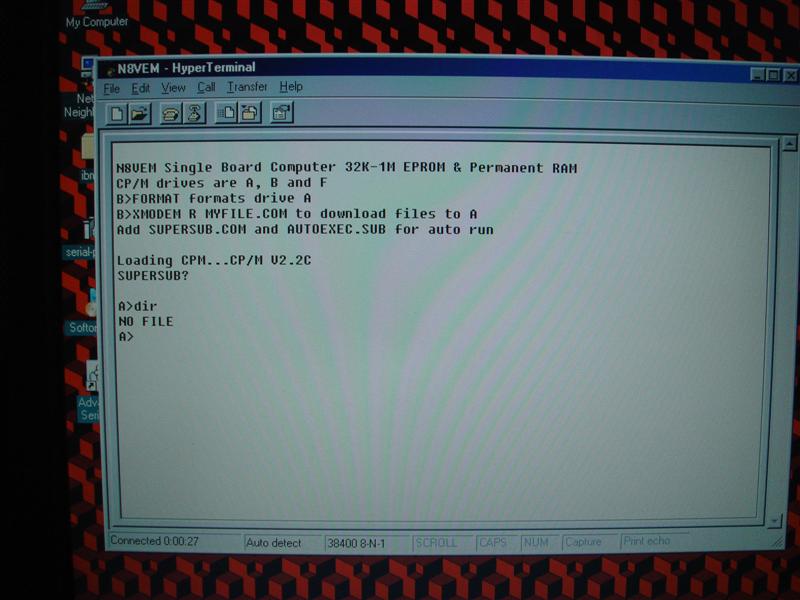
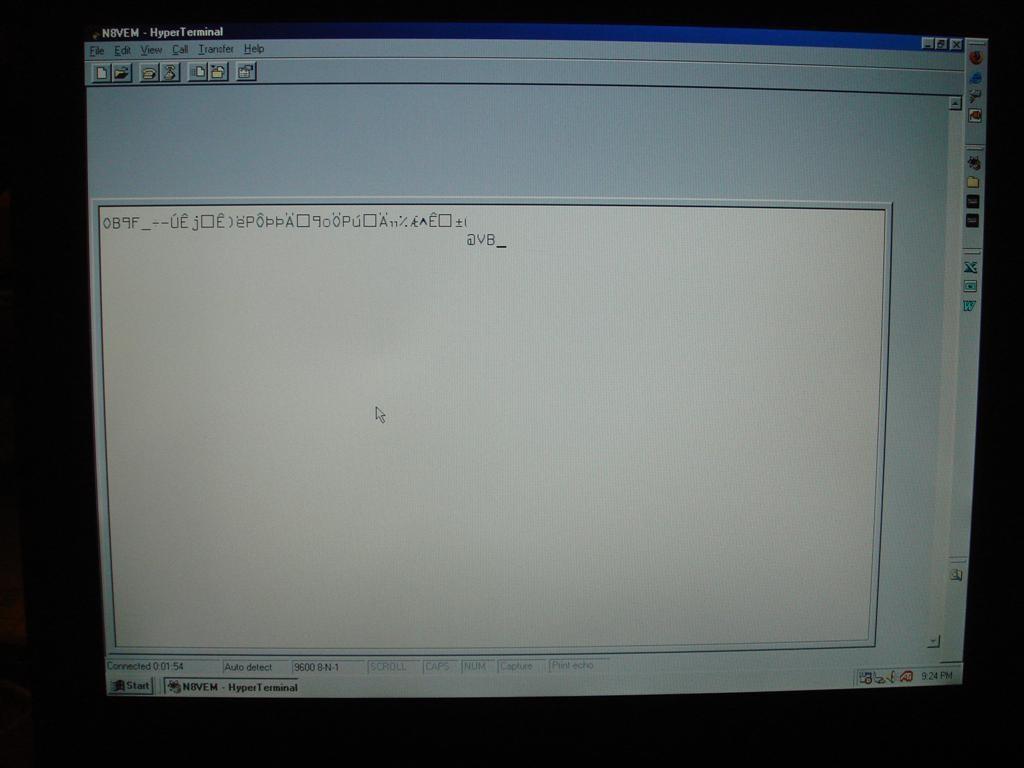
The basic outline is as follows:

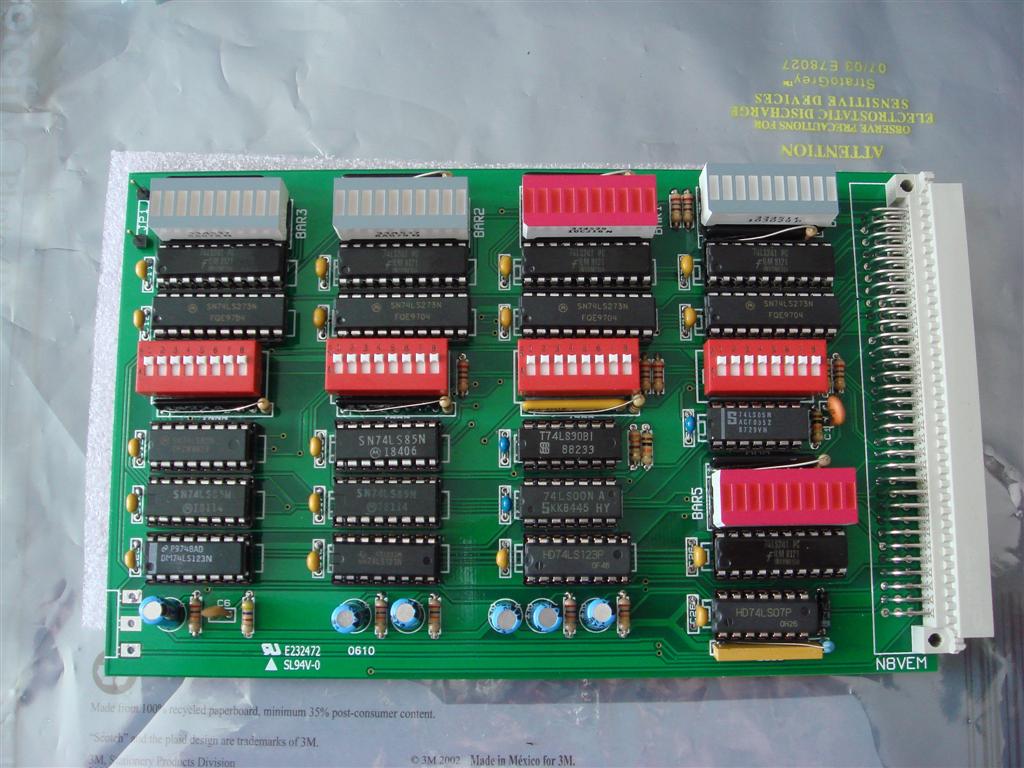
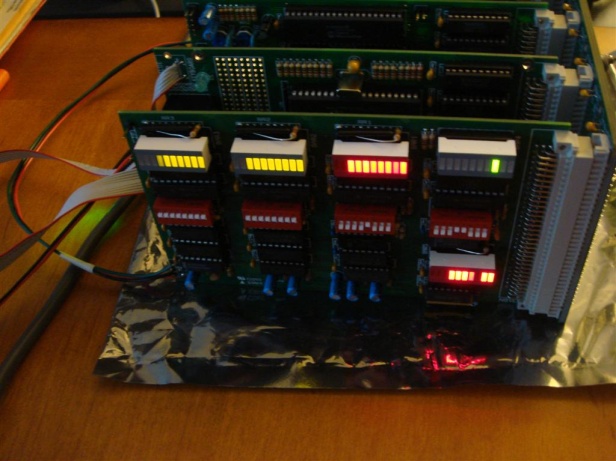
1. order the N8VEM SBC printed circuit board (pcb)
2. obtain the parts list
3. order the parts
4. construct the board and other required elements
5. place the ICs (“integrated circuits” or chips)
6. configure the board
7. test the board
8. next steps

OK, so let’s start!

1. **Order the N8VEM SBC pcb**
   1. OK, so this is the easy part. Andrew Lynch is the originator of this project and he manages the pcbs, so you need to contact him at [**lynchaj@yahoo.com**](mailto:lynchaj@yahoo.com) and tell him you want to purchase an N8VEM SBC pcb.  
        
      When you receive the pcb, it is a good idea to look over it for any obvious defects. Finding and fixing defects earlier is always better than later. One thing to look for is potential “spurs”. These are basically tiny flakes of metal that are a by-product of the manufacturing process that have accidentally fallen onto the board during construction. These look like raised areas under the green insulation layer on the pcb. If they fall on a bad spot they can short out two or more wires by connecting them together erroneously and this can cause major problems. These are generally rare, and quite hard to see, but finding one early and fixing it (by cutting it) can really pay off.
2. **Obtain the parts list**
   1. I presume that since you found this guide, you must have found the wiki site for the N8VEM project. Anyway, it’s here: <http://n8vem-sbc.pbworks.com/w/page/4200908/FrontPage>   
        
      On this site, you can find folders that contain a bunch of documents for each separate pcb that is offered for the N8VEM system. They provide different functions from keyboard and mouse control to VGA output to color graphics and sound to mass storage. Here is where you can get the most up-to-date information about the parts you will need.  
        
      As of April 2011, the N8VEM SBC is at version 2, and the parts I used to build mine are listed in **Appendix A** in the back of this document.
3. **Order the parts**
   1. My main suppliers are Jameco ([www.jameco.com](http://www.jameco.com)) and DigiKey ([www.digikey.com](http://www.digikey.com)). I also use eBay for various items that are not available at either Jameco or DigiKey.
   2. In general, I find Jameco to have better prices and their website easier to use. With DigiKey, their search engine which is really powerful once you know how to use it (and you know exactly what you want) and they have super-fast service. Your experience may vary, though.
   3. Tips:
      1. Make sure you check that you’re ordering the right part! Check the part against the pcb (either the physical pcb sitting in front of you, or a detailed picture of it). This is to make sure the part will fit properly.
      2. Each part will have a DATASHEET on it (usually an Adobe Acrobat (.pdf) file which is available as a link on the website you’re ordering from) which will give you the details about the item, along with detailed measurements
      3. If you’re buying resistors, make sure you’ve got the right resistance (measured in OHMs)
      4. For capacitors, make sure you’ve got the right capacitance (measured in FARADs)
      5. For resistor packs (AKA resistor arrays or resistor networks, abbreviated RR) make sure you get the right OHMs, the right type (BUSSED or ISOLATED), and the right number of pins. Every resistor pack I’ve used on the N8VEM project is of the BUSSED type.
      6. For Integrated Circuits (or ICs, or chips), as long as number of the chip is very similar, you’re usually OK. For example, a **74LS245** chip may be available as **74LS245N** or **SN74LS245N**, which will be very similar, but have very subtle differences. The most important feature of the chips is that they are DIP chips (Dual-Inline Package) – which are used in the N8VEM.   
           
         There is lots of information encoded in the chip ID. For example, in the above example **SN74LS245N**, the “SN” stands for the manufacturer, or Texas Instruments in this case (MC = Motorola, KS = Samsung, etc). The “N” designates the chip as a DIP chip, but many manufacturers do not use this suffix so it’s important to check the datasheet on the chip to make sure you are buying a DIP chip. Even the “LS”, which is the chip type, can be substituted for by “ALS”, “F”, “HCT”, or “AHCT” in many cases.  
           
         If you have a question about parts compatibility, it’s a good idea to search the discussion forum for an answer, and if you don’t find one, ask a question (<http://groups.google.com/group/n8vem/topics>).
      7. IC sockets are recommended for ALL ICs, because they allow you to remove and switch out chips at will, without having to desolder and resolder the ICs on the pcb. Basically, if you’ve got a 14-pin IC you need a 14-pin socket, a 20-pin IC needs a 20-pin socket, and so-on. You solder the socket to the pcb, and then you plug the IC into the socket.
      8. Look at one of the N8VEM pcbs or a picture of one. The solder holes (the tiny holes in the pcb with a thin silvery edge) that are the closest together are 0.1 inches (which is 2.54mm) and the larger distances are usually factors of this (0.2 inches, 0.3 inches, etc). **REMEMBER THIS**. Use this info to make calculate if the parts you order will fit in your pcb. Or you could just measure your spacing with a ruler to be sure.
4. **Construct the board!**
   1. OK, this is the really fun part.   
        
      You should construct your board beginning with the parts with the lowest profile, and move to the parts that have the highest profile. This approach will minimize the need to support the parts from the other side (as the highest profile parts will be held against the board by the surface you are soldering on). My approach generally follows that plan, but I like to start with the chip sockets (which are recommended for ALL chips). My reasoning is that sockets are not temperature sensitive so you minimize the risk of damaging temperature sensitive parts when you do them first.  
        
      First, take the pcb and lay it on a static-free surface (I usually use a medium to large anti-static bag for this) with the “silkscreen” side upwards (the side with the while lettering on it and outlines of the parts). Place the chip sockets in, making sure the notch in the socket matches the notch on the silkscreen outline (usually pointing to the left). Since the pcb is lying on the table, the pins will not penetrate all the way through, but that’s okay. You’re just getting them in place for now.   
        
        
      When they’re all in, place a stiff, flat surface over it that covers the TOP of the all the sockets (a firm piece of card works well, or an old CD works for a smaller pcb). You’re going to use this to flip the pcb over without all the sockets flying off. So grip the pcb and the firm card together (with the sockets sandwiched in-between), and flip the thing over and lie it down on the table again. The pcb should be flipped over now, with the pins from the IC sockets should be pointing up. The pcb is now in soldering position.  
        
      Soldering probably needs its own whole section, but I’m just going to do a flyby. I learned to solder by watching videos on YouTube. There’s an excellent one here: <http://www.youtube.com/watch?v=I_NU2ruzyc4> (“How and WHY to Solder Correctly” by CuriousInventor on YouTube). I used a 15-watt electronics soldering iron with a narrow tip from RadioShack along with 60/40 Rosin-Core Solder (0.032” diameter). I also bought a few small cheap circuit boards (with holes and foil rings around the holes), stuck bare wires in them, and practiced soldering on this setup first to get the timing right before taking the heat to my N8VEM pcbs. I also used a Donegan head-magnifier with 2.5x lenses to do almost all of my soldering. Since you wear it on your head, it frees up your hands. Magnification helps, because you can see the solder liquefy and be sucked into each joint, and you can be sure your joint is good. It’s also binocular, so you have good depth perception. A company called “Quasar” makes an LED lighting system for the Donegan magnifier that is AMAZING. It’s an array of 6 bright LEDs that attach around the periphery of the magnifier lenses, and provides very bright and even lighting when using the magnifier. I love it and find it indispensible now. Using this setup, I’ve never had a cold solder joint and I’m just a beginner, so I highly recommend it. It’s not a necessity, though. There are a thousand ways to do everything and you will find your own way of doing things, too.  
        
      OK. Once the sockets are all in and soldered on, I then put in the other parts. Many of the other parts (resistors, capacitors, etc) have long wires. What I do is bend the wires with a pair of small pliers (because I like nice 90-degree bends) before inserting them into the holes (from the silk-screen side), and when they’re in place, bend the protruding portion of the wires so they’re at about 45 degrees to the surface of the pcb. This bend keeps the part in place while the pcb is upside-down so you can solder it. Here’s an example:  
        
        
      ONE MORE IMPORTANT NOTE: make sure the part you are putting in is the correct part and that it is oriented correctly. This is **VERY IMPORTANT**, because if you use the wrong part in and solder it in, it is a major pain to remove it (by desoldering) and replace it. I tend to check, double check, and triple check everything before it goes into a solder hole.  
        
      After you’ve soldered the part in, take some wire clippers and very gently clip off the excess wire close to the solder joint. Make sure you do this gently, because if you twist the wire as you cut it, you may disrupt or break the solder joint. Some people recommend clipping the wire BEFORE soldering.  
        
      Certain parts (such as diodes, LEDs, resistor arrays, large cylindrical/barrel capacitors, etc.) have a certain direction (or polarity) that they have to go in. Bussed resistor arrays have the 1st pin marked with a dot printed on the part itself. This pin goes in pin 1 of the pcb. Pin 1 for every part is always marked with a square solder hole on the pcb. Individual resistors and non-barrel capacitors can go in any direction. If you’re not sure, remember to check the Datasheet for the part from the website you bought it from (Jameco, DigiKey, or other).   
        
      Note: I have found that it is best to solder batches of parts at a time. That is, I put 20-30 parts in, solder them all, clip the wires, and then do another batch. I’ve found that if I put too many wires in at one time, the protruding wires get in the way of the soldering iron, and soldering becomes more difficult.  
        
      Work your way through all the parts, making sure you’re putting the right parts in the right places, and with the correct orientation. Some parts (like headers –basically rows of pins used for plugging in cables, etc.) may need you to tape them into place so they don’t fall out (on the non-solder side of the pcb) as you solder them.   
        
      Here is the N8VEM SBC board when the soldering is complete:   
      
5. **Place the ICs (chips)**
   1. So now the soldering is complete! Now is the time to place the chips into the sockets. Make sure you follow a picture of the bare pcb (so you can read the silkscreen which has the chip IDs on them) to make sure you’re putting the right chips in the right sockets. I do this because usually the IC sockets will cover up most of the important identifiers printed on the pcb so this is why it’s important to see an image of the bare pcb as a guide.  
        
      You will likely need to bend the pins of the ICs inwards a bit to make them fit into the chip sockets. To do this, hold the chip firmly by the ends without pins, place the row of pins you want to bend onto a hard flat surface, and apply steady pressure until they bend in the desired amount. This way, all the pins are bent nice and evenly. Go slowly, though, because it’s much easier to bend pins “in” then it is to bend them “out”.
6. **Configure the board**
   1. Now you’ve got to place the option jumpers on the board so it functions correctly. The N8VEM SBC V2 pcb has twelve 3-pin headers (labeled K1-K12). The standard jumper positions are in the 1-2 position. In other words, you place a 2-pin jumper over pins 1 and 2, leaving pin 3 open. Remember, pin 1 has the square solder hole. If you can’t tell looking at your pcb, load up a photo of the bare pcb (either yours or one on the N8VEM wiki). The jumper positions and their functions are provided in **Appendix B**.  
        
        
      This is the completed N8VEM SBC v2 board with chips and jumpers inserted:  
      
7. **Test the board**
   1. **Build the interface cable**  
        
      The N8VEM SBC is a single board computer without any video capabilities in its solitary form. Therefore, in order to test it, you will need to connect it to a terminal (which will provide the keyboard input and monitor output). The easiest way to do this is to construct a cable which will provide a serial port (with a D-sub connector or DE-9 plug) which will plug into a “host” computer running a terminal emulation program. This program will allow you to interface with the N8VEM using the host computer’s keyboard and monitor.  
        
      First, you must build a cable with an IDC-10 plug on one end (plastic rectangular connector with 2 rows of 5 pins for a total of 10 pins) and a female DE-9 plug (ie. a serial port plug) on the other end. Here’s a picture of an IDC-10:  
        
        
      The female DE-9 plug is what you will plug into the serial port of your host computer. Serial ports (on the back of the computer) are male ports (ie. they have pins), so the plug at the end of this cable must be a female plug (they have holes). I do not recommend using accessory serial cables to make this connection (such as null modem cables, etc.). This is because many such cables are wired for specific applications, and may not work with your N8VEM. For instance, a serial data transfer cable is wired very differently from a “straight-through” serial cable (1🡪1, 2🡪2, …) which is different from a null modem cable. Because of this cable inconsistency, I prefer to build the complete custom cable from the beginning knowing exactly what pin is connected where and why, and not to use any serial extension cables. The exception to this is using a serial cable that you know is wired as a “straight through” cable (with each pin on one plug connected to each pin on the other plug 1 to 1, 2 to 2, etc).  
        
      The following cable layout shows what is being connected where in this cable:  
        
         
        
       **IDC-10 side** **DE-9 side**  
       2 🡪 4  
       3 🡪 3  
       5 🡪 2  
       7 🡪 6  
       9 🡪 5  
        
      The #1 pin on the IDC plug is marked with an embossed triangle on the plug, and this pin corresponds to pin #1 on the pcb which is marked with a square solder hole (at the lower-right most position of the plug on the N8VEM SBC V2 pcb). The pins on the DE-9 plug are usually marked right on the plug itself in tiny numbers. Strip some wires and solder away. So the above table means **pin 2** on the IDC-10 side must be connected with a wire to **pin 4** on the DE-9 side, and so on.  
        
      The IDC plugs usually take ribbon cables which connect directly to the plug without soldering (by pressing the ribbon cable into the “teeth” on the back of the IDC plug. Once this is connected, you will need to strip the wires on the other end of the ribbon cable and solder them manually to the DE-9 plug according to the above table. There are probably other ways to do this as well.  
      Always double and triple check where you’re soldering something before you solder it. When you’ve built your cable, use your multimeter to check connectivity between each pin on the IDC-10 side and the DE-9 side according to the arrangement above to make sure you got it right.   
        
      In order to test connections in a plug you cannot stick the multimeter lead into the hole (because it will not fit). Instead, take a spare piece of wire, stick it into the hole for the pin you want to test, and then touch the multimeter lead to that wire, as follows:  
         
        
      Once each pin on the cable tests as outlined in the table above, you’re ready to move on!
   2. **Program the EPROM**
      1. To do this, you will need an EPROM programmer. An EPROM is a chip that you can load information onto that will remain there (unlike RAM which gets erased when you disconnect the power). It basically becomes a little read-only “hard-drive” that holds the system files. The only difference is that it is Read-Only Memory (ROM) once it’s plugged into your N8VEM, so it cannot be used to store data like a real hard-drive. EPROM stands for Eraseable Programmable Read-Only Memory, which means it can be erased (with an ultraviolet light EPROM eraser) and then reprogrammed (with an EPROM programmer).   
           
         A UV light EPROM eraser costs about $20 from many sellers on eBay. It erases the EPROM chip by shining a light into the window on top of the chip for a set period of time which erases the chip. The generally recommended erasing time is around 30 minutes (check the datasheet on your EPROM for the manufacturer’s recommendation). This will likely differ between EPROM erasers, due to differences in the power levels of the UV lights.  
           
         An EPROM programmer is a device that plugs into a computer (via a serial/parallel/USB cable), and has an IC socket in it where you plug your EPROM into. You then load a binary image file into your PC, and it sends this image file to your chip, thus “programming” it. You can get these on eBay for around $40. There is some debate about which EPROM programmer is best. I own the Willem EPROM Programmer 5.0e, and while it works, it is somewhat cumbersome to use and I believe there are probably better choices out there. That being said, it is a very popular programmer, so there’s lots of help out there if you get stuck. Each EPROM programmer is capable of programming numerous chip types and to do this, the programmer has numerous different settings (numbers of pins, timings, etc) which have to be matched to the EPROM chip type.   
           
         For the Willem EPROM programmer, you select the chip type from a menu in the software, and it tells you how to set the dip-switches on the programmer device. It shows you a picture of the dip-switch settings, so this is fairly straight-forward. Where it gets tricky is that some chips also require you to change jumpers on the board as well, and this is not documented very clearly. Not setting these parameters correctly can result in damaging the EPROM chip. Also, the Willem programmer requires a serial interface to communicate to the device, AND either a USB cable or a 9v power adapter to provide power to the board.   
           
         Some of the newer EPROM programmers will set the programmer parameters electronically (not requiring the user to set any dip switches or jumpers), and only require a USB to communicate to the programmer AND provide power.
      2. Find a system build/BIOS image to load onto your EPROM chip (which is a 27C801 chip for the N8VEM SBC v2).   
           
         There are many system builds/BIOS images in the N8VEM wiki page in the **“Software information” folder** -> **“Software and ROM Images” folder**. Unfortunately, finding a suitable BIOS can be tricky (because each is built to support different configurations of hardware).   
           
         I recommend the following image for 1st time builders:   
         **Romimage\_Feb28\_2009\_TOOLS\_Zork.bin** (in the “Feb 28 2009 Build” sub-folder)  
           
         I like this image because it contains basic tools and utilities for transferring files to your N8VEM through the terminal program on your host PC, testing your hardware, has a RAM Drive (A:), and has a bunch of cool stuff on the ROM drive (F:) like Zork (a text adventure game), Wordstar (a word processor), Microsoft BASIC (a programming language), etc.   
           
         ***Note:*** *one builder had some difficulties getting the above ROM working and found that on his SBC the boot would suspend at a “Loading CPM …” message. This builder was able to get the following ROM to work:* ***Romimage\_Feb28\_2009.bin*** *(which is in the same “Feb 28 2009 Build” sub-folder).*
      3. Start up your EPROM programmer software, plug in your EPROM programmer, and configure it for your EPROM. Details on configuration and setup are in **Appendix C** for my setup (using a Willem EPROM programmer to program a 27C801 EPROM using a windows system)
      4. Plug in your erased EPROM into the IC socket in the programmer.
      5. Check that the EPROM is empty (select the menu item **“Action”🡪“Blank check”**). It should then display a message after a few minutes verifying your EPROM is empty. If not, put the EPROM back into the EPROM Eraser and erase it for another 5-10 minutes and recheck it by repeating this step.
      6. Load your ROM image into the EPROM programmer software (**“File” 🡪 “Load”**), and “program” your EPROM chip (**“Action” 🡪 “Program/test RAM”**). This can take up to 10 minutes or so with the Willem programmer.   
           
         When the programming is complete it will automatically check the EPROM against the image it just loaded onto it to make sure it’s correct. If it is not, you will get an error message. This verify step should match perfectly, confirming you programmed the EPROM successfully. Once your EPROM chip is successfully programmed, cover the erase window on it with a small opaque sticker. This is to prevent your data from being corrupted or erased by ambient UV light.  
           
         If your EPROM did not program successfully, check all your jumper settings and cables, erase the EPROM, and try to burn the image to the EPROM again.   
           
         Failed EPROM programming can be due to failing to fully erase the EPROM using the UV light eraser. I have found that erasing it for 15-20 minutes works for my setup and chip, but your mileage may vary. Be careful not to “over-erase” which can damage the chip. If you have trouble with this, ask on the Discussion Forum.
      7. **Congratulations!** You’ve programmed your EPROM which now contains the system files you will need to boot your N8VEM. Plug this chip into your N8VEM (making sure you plug it in the right way by using the notch on the top of the chip as a reference and matching this to the socket).
   3. **Turn it on!!!!!**
      1. OK, time to check to see if all the work has paid off.
      2. Set up your host computer
         1. Load your terminal emulation program and check the settings (these are the correct settings for the ROM image I indicated earlier):

**Baud: 38400  
Data bits: 8  
Parity: None  
Stop Bits: 1  
Flow Control: None**

* + 1. Plug one end of the N8VEM serial cable you built earlier into the N8VEM SBC (P3 header) and the other end into your host computer serial port.
    2. Plug in the power source to the N8VEM. When you do this, I recommend watching the board very closely for the first minute, making sure it doesn’t start smoking anywhere and checking the chips and power supply to make sure they don’t get too hot too quickly. If anything alarming happens, disconnect the power source immediately and ask for help on the Discussion Forums.
    3. Switch on the N8VEM! Wait a few seconds, check again that nothing is smoking, and then press the “reset” pushbutton (SW1).
    4. If everything is perfect, the terminal screen on the host computer should display a welcome screen which should look like the following (for the ROM image selected earlier):  
         
         
       If so, **congratulations**! You’ve successfully built a single-board computer, loaded the operating system onto it, and interfaced to it using a cable you built as well! Great job!  
         
       If you see a screen of garbled characters like this:  
         
         
       you still may be okay. The most common reason for garbled characters is an incorrect BAUD setting, so double-check that first, and compare to the BAUD setting that your BIOS is built to work at (usually listed in the description file for the BIOS image you burned to the EPROM). There are other reasons that could account for this, but checking the BAUD setting on the terminal program is the quickest and easiest place to start. I ran into this problem with my board, and correcting the baud speed fixed it. If you still have problems, post a question on the discussion boards. There’s bound to be someone who has seen that problem before!  
         
       Spend some time finding your way about the N8VEM CP/M operating system. With the system build I selected, Drive A: is a RAM drive, which means it can store files (up to 445KB) which will stay there until you power off. Drive B: contains some useful utilities like FORMAT.COM, which formats the A: drive initially, and XMODEM.COM which is used to transfer files to the N8VEM via the terminal program on your host computer. Drive F: contains some fun programs like Zork 1, 2, and 3. It also has Wordstar (one of the first successful word processors), Microsoft BASIC, and a bunch of other utilities as well (like PIP.COM, which is used to copy files).

1. **Next steps.**  
   So hopefully your machine is working! If it is, you can start thinking about what other peripheral boards you would like to add to it to really make it interesting. My favorites are the following:  
     
   The **Bus Monitor**, which shows you the data traveling back and forth on your bus as you use your N8VEM, which is really cool to look at, but can also be used to trap events and do hardware step-through debugging:  
      
   The **PropIO board**, a board that allows you to connect a PS/2 style keyboard, VGA monitor, and a MicroSD card (for mass storage) directly to the N8VEM:   
      
     
   The **Sprite-Color-Graphics board**, or **SCG board**, which provides color graphics, stereo sound, and joystick support so you can play some games, etc:   
      
     
   For you to interface the N8VEM SBC board with almost all of the peripheral board boards, you will need to **ECB Backplane**, which allows you to connect multiple boards together and provides the “backbone” to the system:   
      
     
   There are also boards that allow you to add a hard drive/CF card, SD card, floppy drive, composite monitor, mouse, etc. And when all that’s done, you can work on building a computer case to house your masterpiece!   
     
   I hope this guide has helped you. Enjoy!  
     
   Lok.

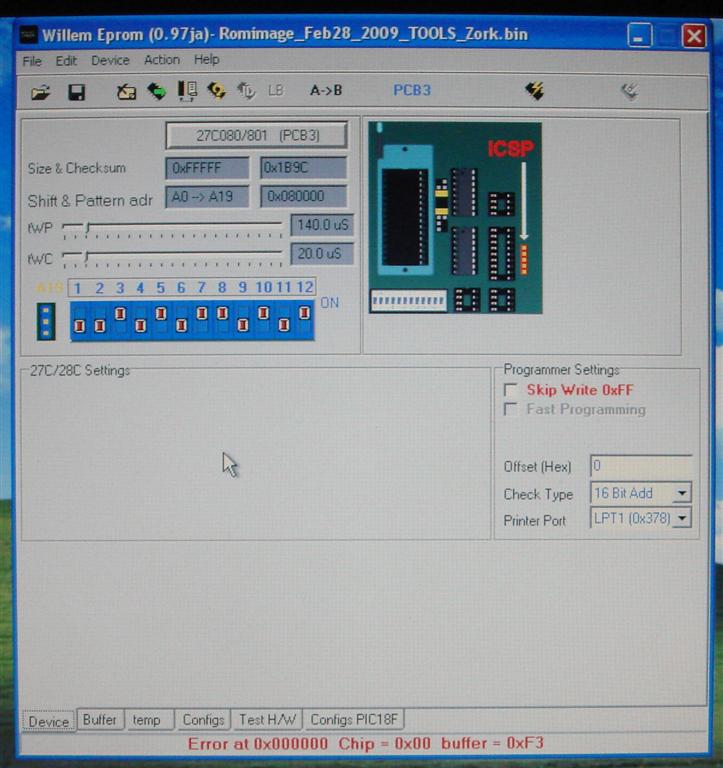
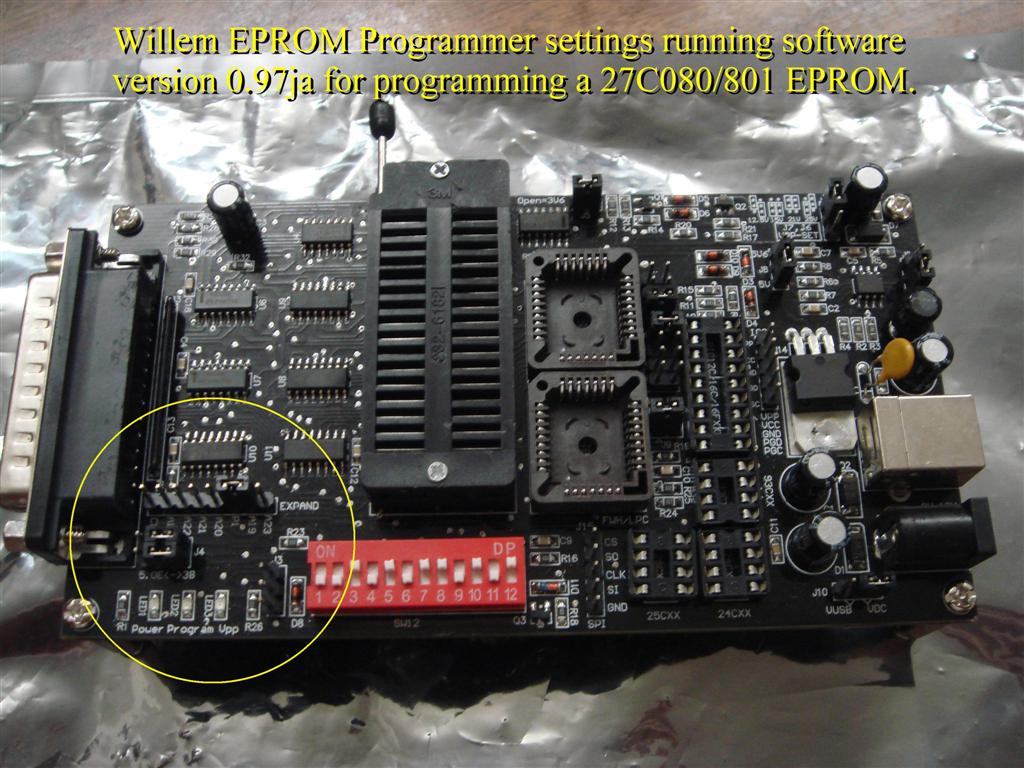
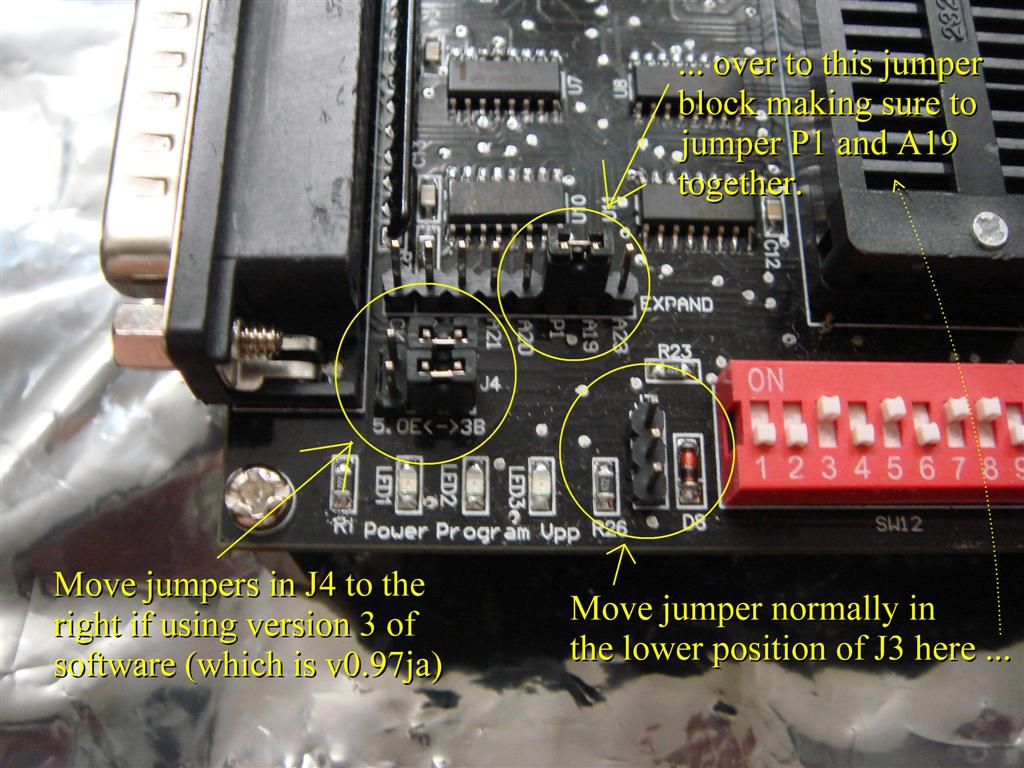
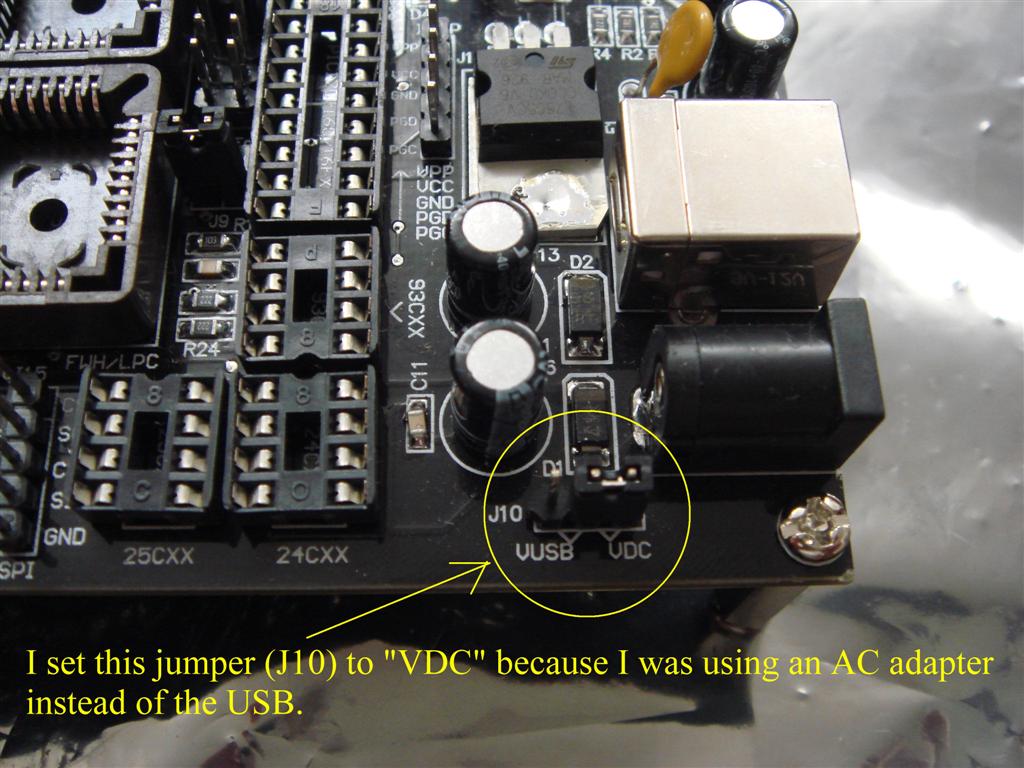
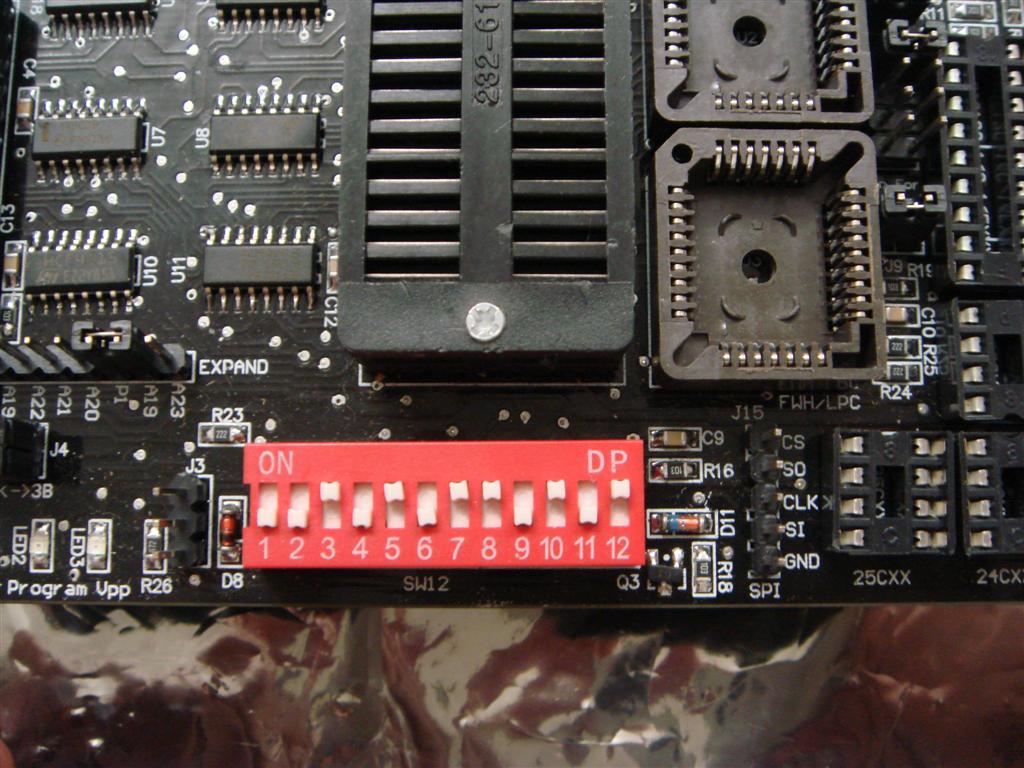
**Appendix A – N8VEM SBC v2 Parts List**

| **Build** | **Board Ref** |  | **Item Description** | **Jameco Part #** | **Digikey Part #** |
| --- | --- | --- | --- | --- | --- |
| 1 N8VEM | BT1 | BATTERY |  |  |  |
| 1 N8VEM | C1 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C2 | 10 uF | C1V5 |  | P13478-ND |
| 1 N8VEM | C3 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C4 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C5 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C6 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C7 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C8 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C9 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C10 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C11 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C12 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C13 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C14 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C15 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C16 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C17 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C18 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C19 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C20 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C21 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C22 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C23 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C24 | 1.0 uF | C1V5 | 545609 |  |
| 1 N8VEM | C25 | 1.0 uF | C1V5 | 545609 |  |
| 1 N8VEM | C26 | 1.0 uF | C1V5 | 545609 |  |
| 1 N8VEM | C27 | 1.0 uF | C1V5 | 545609 |  |
| 1 N8VEM | C28 | 1.0 uF | C1V5 | 545609 |  |
| 1 N8VEM | C29 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C30 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | C31 | 22 uF | C1V5 | 1946308 |  |
| 1 N8VEM | C32 | 0.1 uF | C2 | 25523 |  |
| 1 N8VEM | D1 | LED | LEDV | 253833 |  |
| 1 N8VEM | D1 | JUMPER (LED) | SIL-2 |  | 3M9447-ND |
| 1 N8VEM | D2 | DIODE | D3 |  | 1N4148FS-ND |
| 1 N8VEM | JP1 | JUMPER | SIL-2 |  | 3M9447-ND |
| 1 N8VEM | JP2 | JUMPER | SIL-2 |  | 3M9447-ND |
| 1 N8VEM | K1 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K2 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K3 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K4 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K5 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K6 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K7 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K8 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K9 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K10 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K11 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | K12 | CONN\_3 | SIL-3 |  | 3M9448-ND |
| 1 N8VEM | P1 | PARALLEL | pin\_array\_13x2 |  | 609-3240-ND |
| 1 N8VEM | P2 | POWER | bornier4 | 109146 |  |
| 1 N8VEM | P3 | SERIAL | PIN\_ARRAY\_5x2 | 67821 |  |
| 1 N8VEM | P4 | CLK\_CPU | 8dip300 - 4 MHz |  | XC259-ND |
| 1 N8VEM | P5 | CLK\_UART | 8dip300 - 1.8432 MHz |  | XC231-ND |
| 1 N8VEM | P6 | RESET | SIL-2 | 108338 |  |
| 1 N8VEM | R1 | 10 | R3 | 690380 |  |
| 1 N8VEM | R2 | 10K | R3 | 691104 |  |
| 1 N8VEM | R9 | 100 | R3 (x10) |  | 100QBK-ND |
| 1 N8VEM | R11 | 10K | R3 | 691104 |  |
| 1 N8VEM | R12 | 4700 | R3 | 691024 |  |
| 1 N8VEM | RR1 | 1000 | SIL-6 |  | 4306R-1-102LF-ND |
| 1 N8VEM | RR2 | 4700 | SIL-6 |  | 4306R-1-472LF-ND |
| 1 N8VEM | SW1 | SW\_PUSH | SW\_PUSH\_SMALL | 1586074 | SW400-ND |
| 1 N8VEM | U1 | 16550 | 40dip600 | 288809 |  |
| 1 N8VEM | U2 | 27C080 | 32dip600 | 156275 |  |
| 1 N8VEM | U3 | 74LS08 | 14dip300 | 46375 |  |
| 1 N8VEM | U4 | 74LS10 | 14dip300 | 46404 |  |
| 1 N8VEM | U5 | 74LS125 | 14dip300 | 46501 |  |
| 1 N8VEM | U6 | 74LS139 | 16dip300 | 46623 |  |
| 1 N8VEM | U7 | 74LS14 | 14dip300 | 46640 |  |
| 1 N8VEM | U8 | 74LS245 | 20dip300 | 47212 |  |
| 1 N8VEM | U9 | 74LS245 | 20dip300 | 47212 |  |
| 1 N8VEM | U10 | 74LS243 | 14dip300 | eBay | eBay |
| 1 N8VEM | U11 | 74LS245 | 20dip300 | 47212 |  |
| 1 N8VEM | U12 | 74LS174 | 16dip300 |  | 296-3664-5-ND |
| 1 N8VEM | U13 | 74LS174 | 16dip300 |  | 296-3664-5-ND |
| 1 N8VEM | U14 | 74LS32 | 14dip300 | 47466 |  |
| 1 N8VEM | U15 | 74LS32 | 14dip300 | 47466 |  |
| 1 N8VEM | U16 | 74LS32 | 14dip300 | 47466 |  |
| 1 N8VEM | U17 | 74LS244 | 20dip300 |  | 296-1653-5-ND |
| 1 N8VEM | U18 | 74LS174 | 16dip300 |  | 296-3664-5-ND |
| 1 N8VEM | U19 | 8255 | 40dip600 | 52417 |  |
| 1 N8VEM | U20 | C96ABC | C96ABCMC | 859561 |  |
| 1 N8VEM | U21 | DS1302 | 8dip300 | 176778 |  |
| 1 N8VEM | U22 | MAX232 | 16dip300 | 318959 |  |
| 1 N8VEM | U23 | SRAM\_512KO | 32dip600 | 1927617 |  |
| 1 N8VEM | U24 | Z80-CPU | 40dip600 | 35705 |  |
| 1 N8VEM | U25 | DS1210 | 8dip300 |  | DS1210+-ND |
| 1 N8VEM | U26 | 74LS06N | 14dip300 |  | 296-15069-5-ND |
| 1 N8VEM | X1 | CRYSTAL | SIL-2 - 32.768 KHz | 325244 |  |

**Appendix B – N8VEM SBC v2 Jumper Settings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Board**  **Reference** | **Jumper Description** | **Jumper Setting** | | **Notes** |
|  |  | **1-2** | **2-3** |  |
| **JP1** | **battery backup** | **x** | **n/a** | **JP1 installed allows you to use the common battery backup on the ECB bus (pin A24). Not installed means either the local battery is used or no battery backup at all if no battery.** |
| **JP2** | **One bit input port** | **x** | **n/a** | **JP2 installed enables the one bit input port. This is currently unused although could be used for external input if desired like a button or other things. Not installing JP2 disables the one bit input port.** |
| **K1** | **U2 EPROM chip pins (32-pin or 28-pin)** | **x \*** |  | **32-pin EPROM used in U2 (ex. 27C080 1MBx8 EPROM) - default** |
|  |  |  | x | 28-pin EPROM used in U2 (ex. 27C256 EPROM) |
| **K2** | **UART side hardware handshaking (DSR or CTS)** | **x \*** |  | **DSR (this should be paired with K3 - DTR) - default** |
|  |  |  | x | CTS (this should be paired with K3 - RTS) |
| **K3** | **UART side hardware handshaking (DTR or RTS)** | **x \*** |  | **DTR (this should be paired with K2 - DSR) - default** |
|  |  |  | x | RTS (this should be paired with K2 - CTS) |
| **K4** | **serial side hardware handshaking (DSR or CTS)** | **x \*** |  | **DSR (this should be paired with K5 - DTR) - default** |
|  |  |  | x | CTS (this should be paired with K5 - RTS) |
| **K5** | **serial side hardware handshaking (DTR or RTS)** | **x \*** |  | **DTR (this should be paired with K4 - DSR) - default** |
|  |  |  | x | RTS (this should be paired with K4 - CTS) |
| **K6** | **U2 chip type (27C080 EPROM or 29C040 flash)** | **x \*** |  | **27C080 EPROM used in U2 - default** |
|  |  |  | x | 29C040 flash used in U2 |
| **K7** | **U23 SRAM (512K or 128K)** | **x \*** |  | **512K chip used in U23 - default** |
|  |  |  | x | 128K chip used in U23 |
| **K8** | **U2 chip type (27C080 EPROM or 29C040 flash)** | **x \*** |  | **27C080 EPROM used in U2 - default** |
|  |  |  | x | 29C040 flash used in U2 |
| **K9** | **Parallel Port power status (pin 25)** | **x \*** |  | **GROUND - default** |
|  |  |  | x | VCC (\*\* WARNING \*\* - this setting will send VCC power down pin 25 of parallel port which may potentially damage connected device) |
| **K10** | **MCPL (Memory Page Config Latched)** | **x \*** |  | **32K upper RAM fixed/32K lower RAM switchable memory map - default** |
|  |  |  | x | 48K upper RAM fixed/16K lower RAM switchable memory map (banked) |
| **K11** | **MCPL (Memory Page Config Latched)** | **x \*** |  | **32K upper RAM fixed/32K lower RAM switchable memory map - default** |
|  |  |  | x | 48K upper RAM fixed/16K lower RAM switchable memory map (banked) |
| **K12** | **Bus Interrupt (pin A23)** | **x \*** |  | **internal UART interrupt - default** |
|  |  |  | x | external ECB interrupt |
|  |  | **\* = default setting** | |  |

**Appendix C – Willem EPROM Programmer Configuration**

1. Use **Willem Software version 0.97ja** (version 3) – this is because there have been some internet documented cases of later software versions causing some unusual problems with certain chips (including 27C080/27C801 which is used on the N8VEM). It should be on the software CD your programmer came with. It not, there is a copy on the N8VEM wiki page. Load up the software on your host PC and it should look like this:  
     
   
2. Plug in your Willem EPROM programmer to your parallel cable and plug your parallel cable into your PC. Also plug in your power source (either a USB cable or an AC Adapter). Some of the LEDs on the Willem EPROM programmer should light up.
3. Set up the jumpers on your Willem EPROM programmer board as follows:  
     
     
     
     
     
     
     
     
   Check the Willem software to tell you how to set these DIP switches:  
   
4. In the Willem software, select **“Help” 🡪 “Test hardware”**. At the bottom of the window, there should be a message **“Hardware present”** which means it detected your hardware properly and you are ready to program an EPROM!  
     
   If the message says **“Hardware Error”**, check all your cables and settings and repeat.