Some basics rules for hobbyists which are playing with PICmicros

1. Eight steps to achieve a functionaly embedded project

Most of the hobbyists are impatiently when they start working with microcontrollers. This leak of patience is often translated in using quick and dirty methods for finalize their project. Something that works on the table, like a solderless breadboard [1] covered with wires and with a bunch of components hanging in the air, but stop working if someone sneeze around can't be named a project but just a test. Hopefully is a fully functionally test. Paying the same attention to all of the project phases is the key for a good result. Those phases should be:

a) **Defining and understanding the goal of the project** is essential. Assign some time for this operation, search other similar projects on the WWW (World Wide Web) and inspire from those but do not copy them 1:1 even you're a beginner. Your own creativity must always be visible in your projects. Save your documentation in well organized folders so you can easily find later. Print on paper things you don't understand for studying them. Let the time lay over your initial thoughts until everything is crystal clear in your mind (this could be really difficult because of the curiosity or indecisive attitude).
Tips: Searching for answers to a specific problem about embedded design will bring you information about other related issues. Do not ignore them but save those, will be useful later.

b) **Choosing the right microcontroller and all other components** based on **a)** but also:
- check the availability of development tools (compiler [2], programmer hardware+software [3], editor [4], maybe a simulator [6]) at your fingers
- verify your own hardware skills like soldering SMD (Surface Mount Device) or TQFP (Thin Quad Flat-Pack) packages [7] or manufacturing boards,
- verify your programming skills level and the availability of examples and compatible libraries [5] for the chosen compiler
- check the physical availability of your microcontroller at the vendors from the geographical area where you live [8] **Tips:** For a real hobbyist, the price of the microcontroller is not essential, but buying one from the other side of the world could not be very funny. Do not order microcontrollers which aren't in vendor's stock but on the sale list only, it took too much time until arrive.

c) **Download datasheets for all the components involved in the project** as chosen in **b)** read them and print the most important information (like packages dimensions, their footprint and programming specifications). Even if you'll manufacture only one piece of "something" based on this project, prepare your mind as the product will be manufactured in 1000 pieces. In this way you'll avoid a lot of mistakes which else can easily occur. **Tips:** Every component from your project have the same significance. A resistor can be as important as the microcontroller. Read the resistor datasheet as well (and you'll discover interesting things if your resistor is a SMD one or a power one with a cooler on it). Take care to special connectors, identifying which is the front and which is the back end .

d) **Design the hardware part of your project:**
- choose a CAD [9] (Computer Assisted Design) software, if you don't already have one,
- define newer components into the CAD library and double check them,
- edit the schematic, (don't rush, you and the others must understand it next year)
- correct the mistakes in the schematic and in the CAD library, use as many sheets as necessary for writing an intelligible schematic, do not use connectors for all component symbols just because has the right number of pins

- verify all footprints of the used components (including those which are already available in the CAD library) and the drill size,
- choose the right layerstack [10] and padstack [11] ; for hobby purposes that's a two layer stack, with through hole pads diameters of minimum 60-70mil (mili inch) or 1.5-1.7mm and minimum drill diameter of 24mil or 0.60mm
- design the PCB (Printed Circuit Board) [13], don't rush, component dispersion on the board is the most important step and determine the routing process simplicity
- verify the PCB as many times you can, understanding the advantage of the netlist inspection
- export the factory files (gerber and excellon) and verify those again using a gerber viewer [12]

Tips: do not let the autorouter to route your traces, use planes for both ground and power nets, route manually all critical signals as power, ground, clocks or analog traces. Choose pads, vias and drill sizes, route and isolation width, all coresponding with your manufacturing technology [13]. If a factory will manufacture your board, do not start routing until you have talk with the guy from the factory and you are absolutely sure that you have understood completely all requirements (including every word of the terminology they use).

e) Manufacture and test your hardware. Use for manufacturing any of your favourite methods, but choose professional methods; for hobby can be thermal transfer [14], inkjet printing [15] or CNC (Computer Numerical Controlled) routing [16]. **Tips:** For boards without solder mask provide enough isolations between pads and copper, routes or vias. Etch (or mill) the copper in the copper plane between the pads which belongs of the same SMD component. The ring of through hole vias and through hole pads should be large enough to assure the mechanical rigidity of the wire/component once is soldered. Test points are mandatory. A complex SMD board without any testing points reveal the low experience level of the board designer. Once the board is designed it must be easily tested. ICSP (In Circuit Serial Programming), power and all signal connectors should have easily access. Since one connector must be taken off to access the nearby connector, that's a sign of bad design.

f) Write your firmware code, program and test it. Split your code in many files related to: microcontroller definition, hardware definition, various library with specific functions as math, communications, LCD (Liquid Crystal Display), etc. Start testing from simple code (procedures) to complex code (libraries, main programs). Do not try to write the whole program at one time. Comment all time critical stuff and other weird constructions or component specific routines. Choose the right method for generating and programming the hex file in your project. Use bootloaders [17] [18] if you expect a large numbers of programming sequence in your microcontroller and you can use a bootloader. Use ICSP programming if your major goal is communications, both UART (Universally Asynchronous Receiver Transmitter) or USB (Universal Serial Bus) are used and you dislike any of 1wire bootloaders [19]. Choose the right method for debugging your project. Use a professional debugger [3] or just the LCD or terminal [20] debugging methods. Sometimes debugging using one LED (Light Emitting Diode) can be enough. Use simulators [6] only for small pieces of code which does not use large software delays. **Tips:** Never expect that someone's firmware (or library) will work on your hardware from the first try, without any problem. Write your code thinking that the final goal is a running microcontroller into a hardware board and not presenting it at a software code exhibition. However, if you understand your previous written code after at least one year, it's a sign you've been a good writer. If you're able to modify a piece of own code after a few years, then compile and load into the same board you have previously designed and it works from the first try, then you probably are an expert.

g) Save your work. Save your project entirely as an archive (source code, hex file, compiler, documentation of everything, IDE, schematic, PCB and fabrication files) at least in two different places, one being a compact disk.

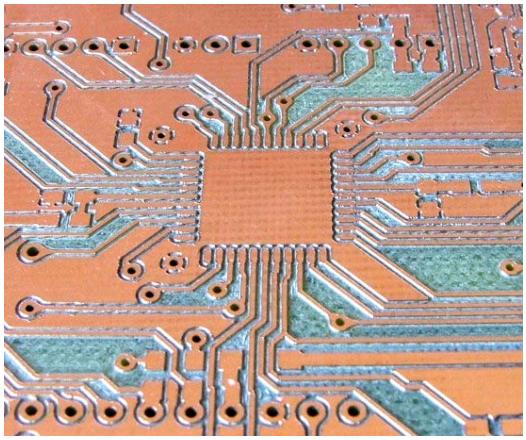
h) Now the project works, what to do ? Throwing away to the garbage and start another one? You can do exactly what you want to, but do not expect to become rich and famous just because you have designed and manufactured a board with a working microcontroller on it. This is a movie happening. **Tips:** Verifying embedded

boards is not easy. Even after two hardware revisions and dozens of software revisions you will still find mistakes and bugs. Don't rush declaring: this project is ready for customers or for being published on the WWW.

2. High resolution footprints give you headaches?

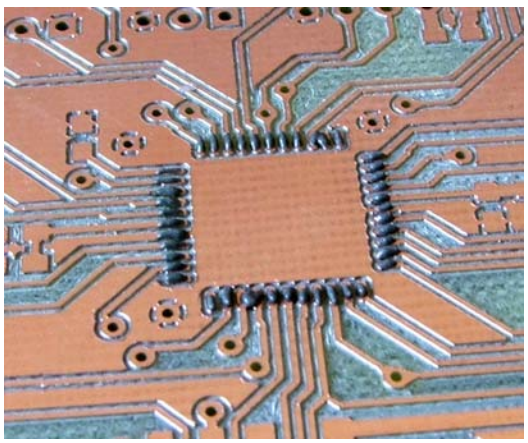
The favourite microcontroller's footprint for hobby is DIL (Dual In Line), 150 or 300mil wide, only because can be used on solderless breadboard or fit easily in a socket. The existence of ICSP makes the socket an obsolete device, so why don't use a TQFP package as well? Using good tools for such operation is mandatory (fig. 1). You need one no clean solder paste syringe [21] (upper) with a syringe needle on top, a high quality tweezer (middle) and a sharp needle (below) mounted in a handle tool.

fig. 1 Tools for soldering TQFP



The PCB must be cleaned (fig.2) with a low durity sandpaper (800 or 1000) followed by izopropyl alcohol (or even ethyl alcohol). Do not touch your fingers on the surface after cleaning, use a pair ofcotton gloves for the rest of the procedures. Solder paste must be kept in refrigerator if has less than 6 months since production date or in deep-freezer if has more than 6 months back. Is still usable for about 3-4 years, even the producer does not guarantee it more than 6 months, but after 6 moths must be used cold, so do not keep it at room temperature only when dispense the paste.

fig. 2 Cleaned PCB, manufactured by CNC



Apply gentle a small amount of solder paste longitudinally on the TQFP pads trying to not short circuit the adjacent pads (fig.3). If you've droped too much solder paste, do not scramble. Use the needle from fig. 1 and remove the excess of solderpaste between shortcircuited pads. Do not waste too much time with this procedure because solder will connect anyway the pads when drop the package.

fig. 3 Apply solder paste on footprints

Place carefully the TQFP package, taking care of the pin1 position. Using a good tweezer is mandatory because you need to place it above the solder paste from the first move (fig.4). If you failed and the solder paste has been spreaded on the PCB, clean it using a paper napkin and repeat the procedure from the fig.3. However try to do not make mistakes, because the solder paste on the milled or etched sections (which are deeper than copper) can't be remov easily only by hot air. An acceptable appearance at the end of this step appears in fig.5.

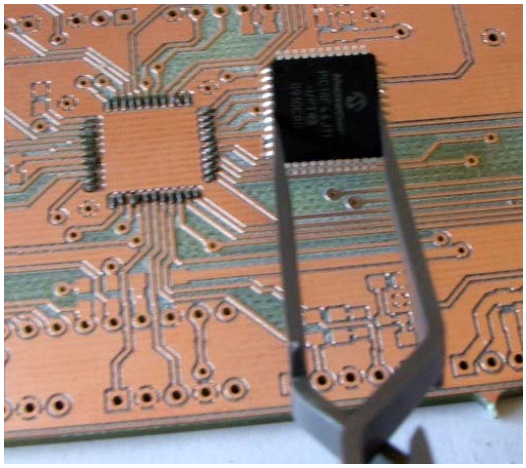
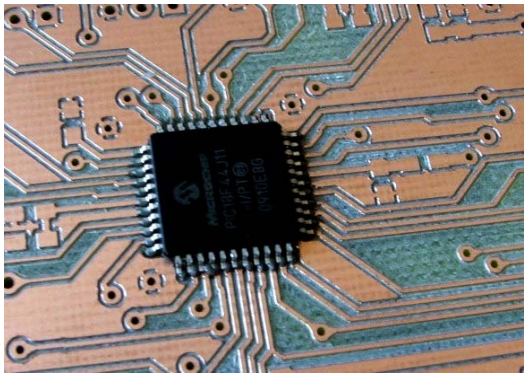
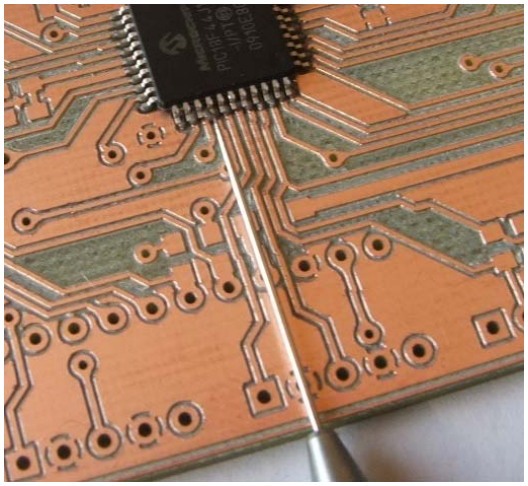


fig. 4 Place the TQFP on the footprint



Solder paste must not be in excess. You can see an excessive amount of solderpaste between pins 3 and 4 (numbering from left to right) on the bottom row of the TQFP (fig.5)

fig. 5 TQFP placed above the solderpaste

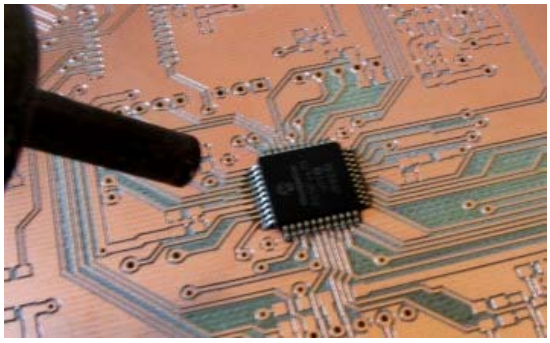


Remove the solderpaste in excess using the needle tool (fig. 6). Set the hot air temperature (fig. 7) to achieve the most appropriate temperature for soldering profile as is described in soldering documentation of the TQFP. Maximum temperature should be around 220C-225C on the PCB surface for the solder paste described. However as large are the contiguous copper planes on your PCB, as higher temperature above those values must be set. Check also the melting temperature of your solder paste, RoHS solder paste have the melting temperature between 180C to 210C. A sign that your hot air temperature is too high at the copper surface is changing the copper color from orange-brown to blue-red.

fig. 6 Remove the excess of solderpaste

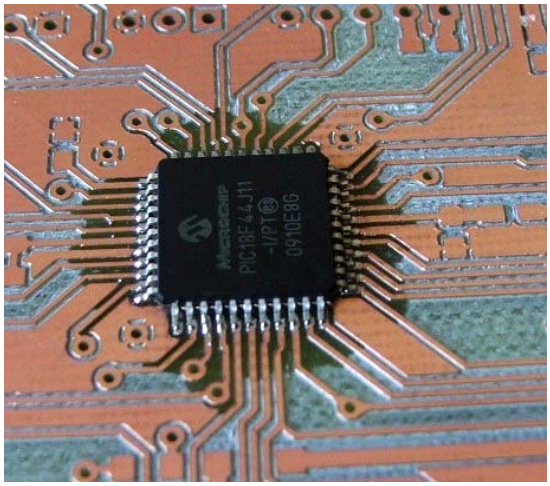
Experimental tests with Chinese KADA hot air rework station shows an optimal air temperature at the output of air gun of around 300-330C for a soldering time less than 20-30 seconds, on a dual layer PCB with copper plane on both layers and ½ oz of copper PCB.

fig. 7 Set the temperature of hot air tool in 250C to 330C range and small air flow value



Solder the package without heating directly the center of the TQFP package, hot air must be directed to the PCB and pins. Time requested for solder the whole PIC18F44J11, TQFP package using a 5mm diameter round tip for the air gun should be less than 1 minute. Once the soldering process begun, the flux content in the solder paste will be spread on the PCB and the solder paste will change the color from gray to shine silver.

fig. 8 Solder the microcontroller



At the end of the soldering process all solderpaste must be melted. WARNING: if you don't solder it completely, short circuits will appear. A sign that excessive solderpaste was used is the size of the blotch made by no clean flux on the PCB, which will be more than 50% of the package size on all directions. Since is a no clean flux, it does need to be cleaned.

fig. 9 The aspect of a resonably soldered package

3. Refecences

[1] Solderless breadboad aspect at http://en.wikipedia.org/wiki/Breadboard

[2] Kyle York, free JalV2.4m compiler: http://www.casadeyork.com/jalv2/

[3] Microchip Pickit2 programmer and debugger, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805

[4] Sunish Issac, free Jaledit 07.20: http://code.google.com/p/jaledit/downloads/list

[5] Rob Hammerling, Joep Suijs, Sebastien Lelong, Matthew Schinkel, Albert Fabber, Eur van Andel, Rickard Zengerik, William Welch, Michael Watterson, Stef Mientky, Javier Martinez, Wouter van Ooijen, free libraries for JalV2: http://code.google.com/p/jallib/

[6] Vladimir Soso, free PIC simulator, http://www.oshonsoft.com/pic.html

[7] Microchip PIC packages specifications: http://www.microchip.com/packaging

[8] Some huge PIC microcontroller vendors:

- Easten Europe: TME http://www.tme.eu/en/
  Comet http://www.comet.srl.ro/main/index.html
- Europe: Future Electronics http://www.futureelectronics.com/en/Pages/index.aspx
  Conrad http://www.conrad.com/
- USA: Digikey http://www.digikey.com/
  Mouser http://eu.mouser.com/Home.aspx

[9] Some freware or free CAD software:
- Eagle http://www.cadsoft.de/freeware.htm
- KiCAD http://iut-tice.ujf-grenoble.fr/kicad/

[10] PCB layerstack standard: http://www.pcbmatrix.com/downloads/files/PCB%20Layer%20Configurations.zip

[11] PCB padstack standard: http://www.pcbmatrix.com/downloads/files/Padstacks.zip

[12] Free gerber CAM wievers: http://www.mitsi.com/PCB/free%20viewers.htm

[13] PCB design for manufacturability rules: http://www.pcbmatrix.com/downloads/files/GenDocs/Design%20For%20Manufacturing.zip

[14] Thomas P. Gootee, Laser printer toner transfer: http://www.fullnet.com/~tomg/gooteepc.htm

[15] Stefan Thretian, Direct Inkjet Resist Printing: http://techref.massmind.org/techref/pcb/etch/c84-st.htm

[16] CNC for hobby, http://groups.yahoo.com/group/CNC-PCB_Design/messages

[17] Claudiu Chiculita, tiny PIC bootloader http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm

[18] Rob Hamerling, Tiny Bld2, http://www.robh.nl/

[19] Wouter van Ooijen, Wloader http://www.voti.nl/wloader/

[20] Serial Terminal http://realterm.sourceforge.net/

[21] R276 Kester dispensable no clean solderpaste http://media.digikey.com/pdf/Data%20Sheets/Kester%20PDFs/R276_Global_(21Sep09)[1].pdf

Note: last visited links above: 01.04.2010

4. Vitae

The autor of this paper is a 45 year old senior engineer, since 1986 is involved in hardware research and electronic design at the National Institute for R&D for Isotopic and Molecular Technologies, Cluj-Napoca, Romania, Eastern Europe. He designed over 100 different projects including analog, digital, mixed signal, embedded and RF applications. His most complex mixed embedded designs include a 12 layer PCB using Virtex 5, Stratix II, fast multicore DSP running at 3x600MHz and 5GHz transcievers. He is still designing hardware prototypes and programming with JAL which add some fun.