# PICBOT PCB - TWO

## Introduction

This should be titled "What I Learned from the first PicBot Prototype", describing how I would do it now that I've made enough mistakes to know better (not best perhaps, but better).

### MAJOR DIFFERENCES FROM FIRST PROTOTYPE

***Separate PCBs for Separate Functions***
The Main PCB handles I/O, but shuns complex stepping motor controls, which could be done by a more specialized board. Instead servos, digital I/O and Analog inputs are handled here. Data is passed through the SCI to the host PC, while "sister" boards, such as the motor controller or LCD's, are controlled from this board via the SPI port (of which I'm planning to have three). This also cuts down on the number of headers crowding the main board.

***Ease of Construction***
A major difference is the use of resistor packs for pull-downs (or -ups), saving much complexity, lots of space and some seriously ugly construction (all those resistors soldered to the bottom, bridging other traces--yuk!). I won't do that again.

***Motor Controller PCB***
A separate stepping motor controller board, a "sister" board to the main PicBot PCB, could employ a much simpler MCU (such as the PIC 16F84 or even the 8-pin 12F629) to handle simple input signals to output step motor phases. It would also have the standard driver IC as well to handle the power requirements. Using the SPI port for control (as a slave board) it would free up 8 full I/O pins from the PicBot board.

***Pull-Ups on Input Pins***
Decided finally to go-with-the-flow and change Port B to pulled up, active-low inputs. It is a bit easier using ground as a signal instead of passing Vcc to the controls; but the main reason is the easier connections via a resistor pack on the PCB (a minor issue if I decide to change it back due to power efficiency problems).

***Removal of LVP***
And, finally, the removal of the Low Voltage Programming (via the 2-bit setup) from this board, since it should only ever be used once per chip (now that the bootload program has also been redesigned). Instead, a BLIP board (Boot Loader Initialization Programmer) will handle this.

### FUTURE DEVELOPMENTS

***Specialized PCBs***
For a possible future profit, after this prototype, several more prototypes could be created using the same basic design ideas, but different "features". This would mainly have to do with the number of I/O pins and which ones are Analog versus Digital, etc. PCB size, memory, specialized I/O for certain sensors, PC hosting and not--these are all possible "flavors". The inclusion of "sister" boards also makes this a handy idea (if others haven't already filled the market with these).

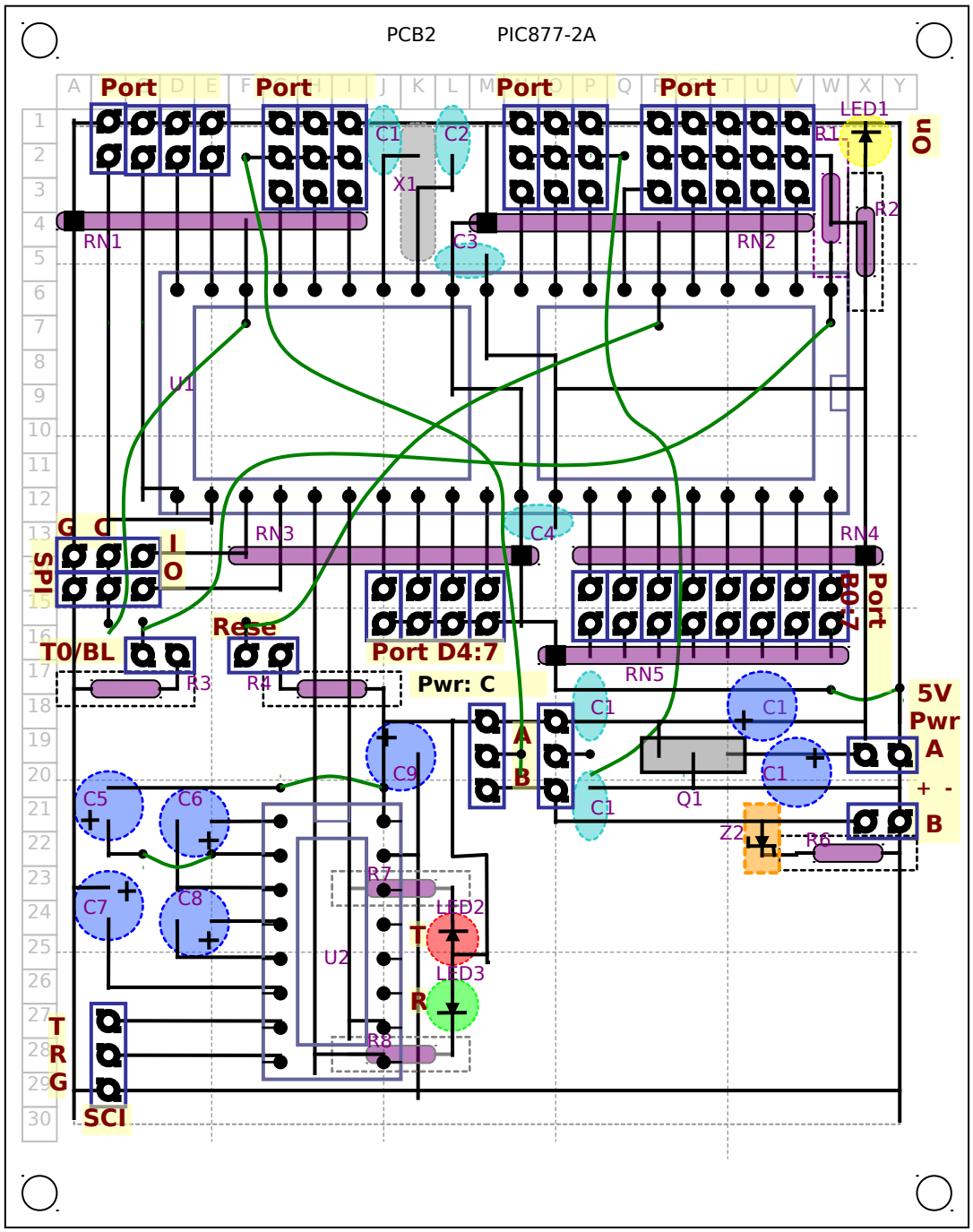
Also, BLIP boards for each type of PIC (or at least the 16F87X chips I like to use).

# Table of Contents

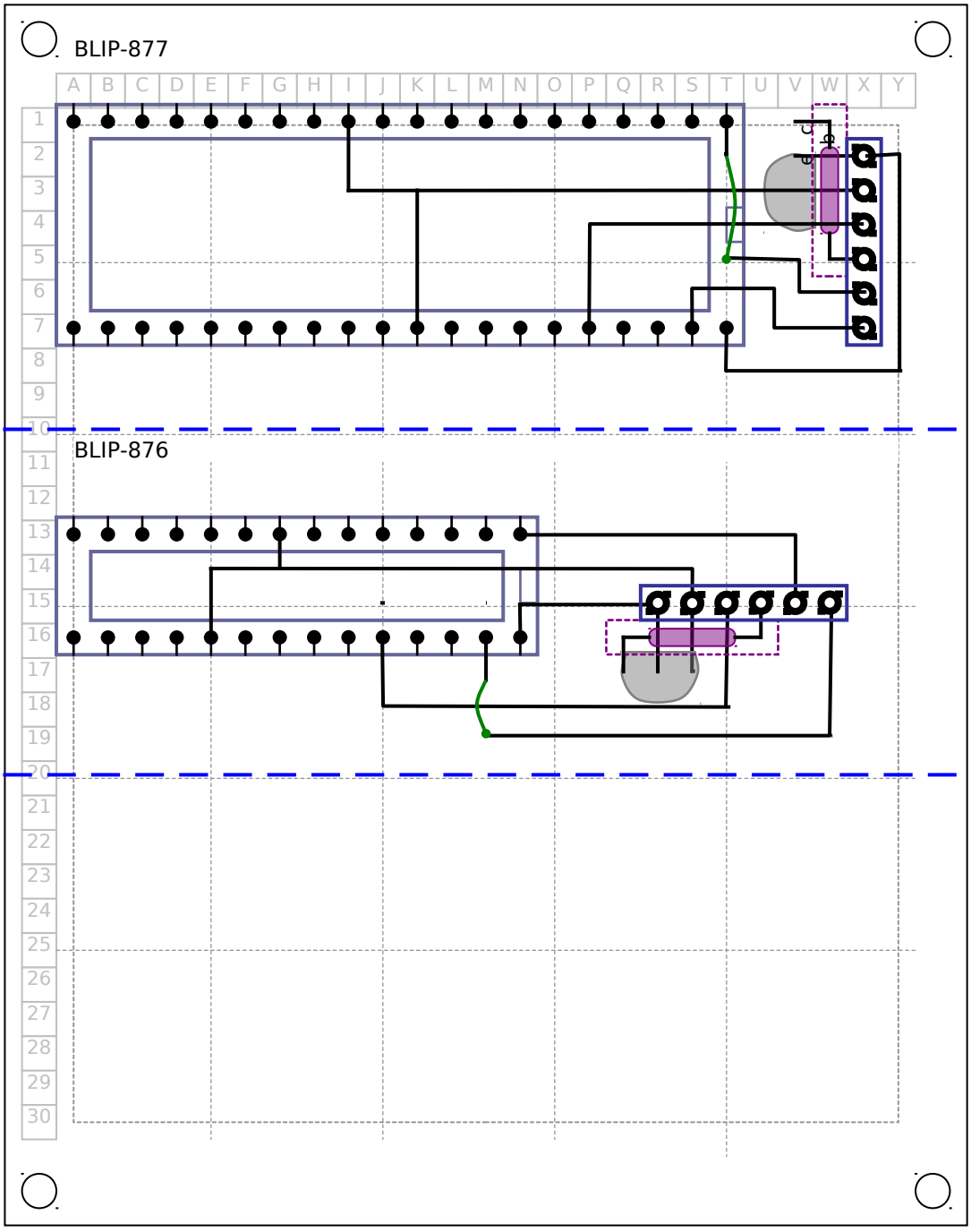
# PCB Layout

## PROTOTYPE – TWO (A)

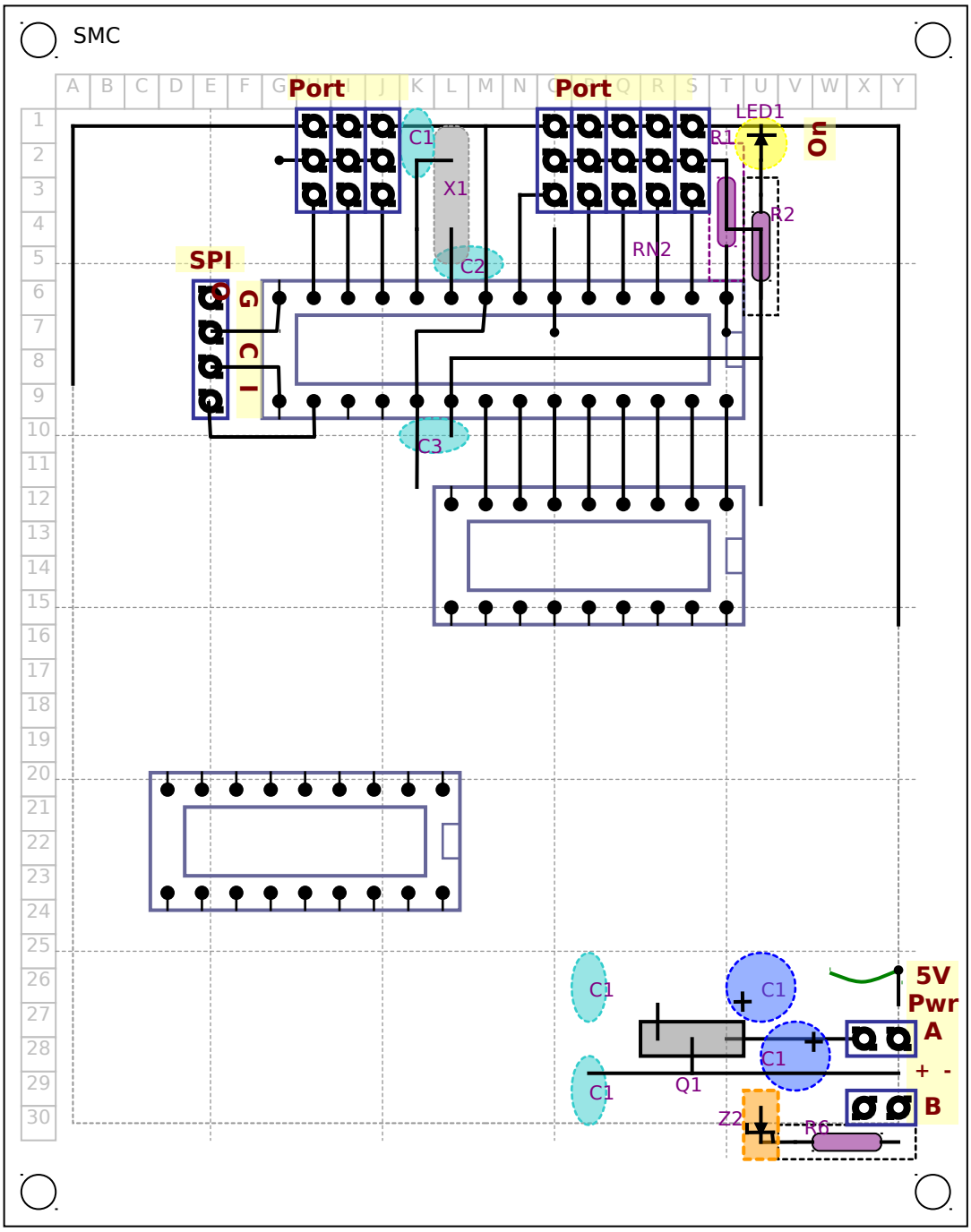
***PicBot-877***



***BLIP (Boot Loader Initialization Programmer)***

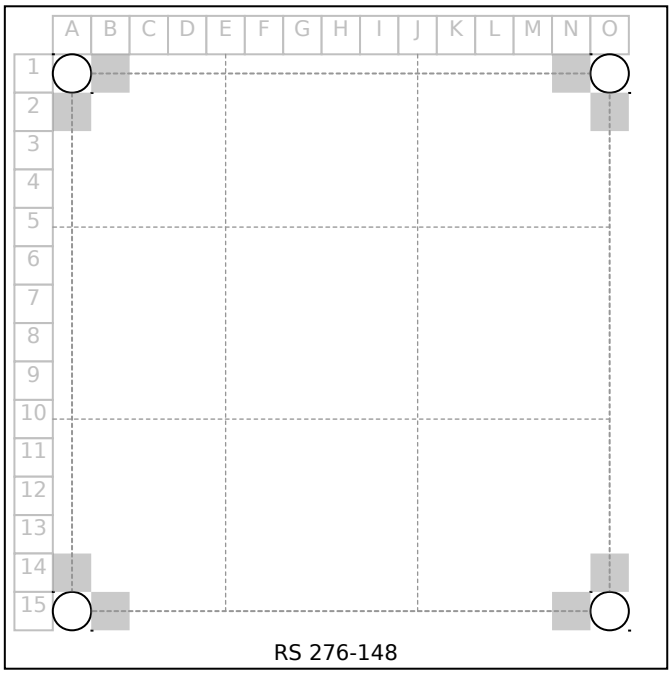


***SMC (Stepping Motor Controller) So Far . . .***



Notes from the future…

- If you decide to go ahead with the SPI-connected-separate-PCB architecture, then it seems to make more sense to not use the PIC 16F877 and go with the 876 throughout.
  - I have yet to find a use for the parallel (D/E) ports, nor should I need SO MANY I/O ports in one chip, especially when separate PCB's are being used anyway—each of which could have its own 877 on it.
  - The size of the 877, along with the use of resistor networks, could extensively shrink the required PCB platform, such that a much smaller PCB (though a little bigger than the Radio Shack 15x15 prototype board pictured below) could easily be used, with far fewer traces and wires needed.
  - See the Peripherals document for the new SPI jumper design, allowing for varied/multiple PCB connections. One variation on this worth looking into:
    - Parallel SPI stacking, in which instead of a round-robin approach, all boards share the same "bus" of lines. The master board sends out an address, then an instruction and only addressed board responds with data, then a terminator.
    - This revised design would allow stacking boards one atop the other (given the proper connectors on both sides of each board, except the master, which has only the one.
    - This would require all <u>other</u> connections would have to be accessible to the <u>side</u> of each board (which I have had problems with in the past breaking from the surface…). Alternativs:
      - Make SPI the <u>only</u> side-connectors and create a long, thin "backplane" PCB with 4-pin connectors mounted at regular intervals to plug each new board into.
      - Do the same with a thin portion of the Master board.
      - Place two 4-pin plugs on each slave board for connecting up the "next" slave board. This is similar to the "daisy-chained" approach except that everybody is sharing the same bus of signals (both plugs on the salve boards share all four wires).



RS 276-148

## THE MASTER BOARD IDEA

This is a possible future version using the 16F876 (for simplicity), driving from zero to seven peripheral boards via an SPI bus, muxed by an 74HC42 (bcd to decimal decoder). What remains of the original prototype are 5 A port pins (AD, I/O, Servo), all 8 B port pins (I/O, Ints), the SCI to talk to a host (PC presumably) and a regulated 5 volt input. One assumes any further interfacing (motor, LCD, etc.) will be done by peripheral “slave” MCU boards.