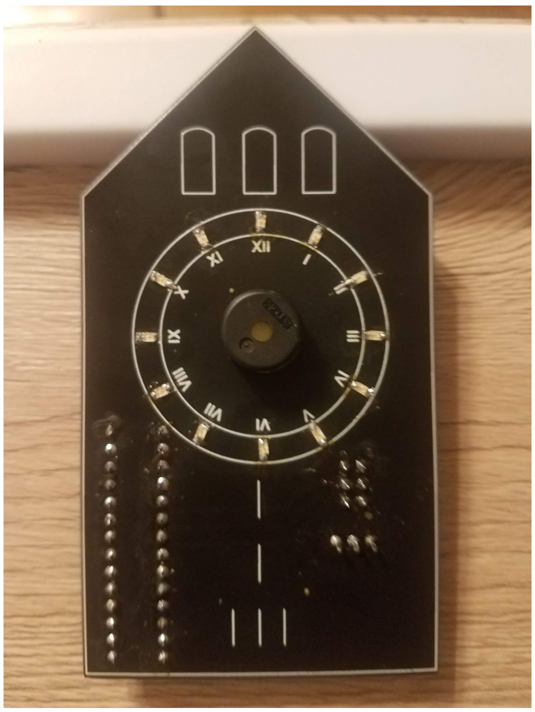
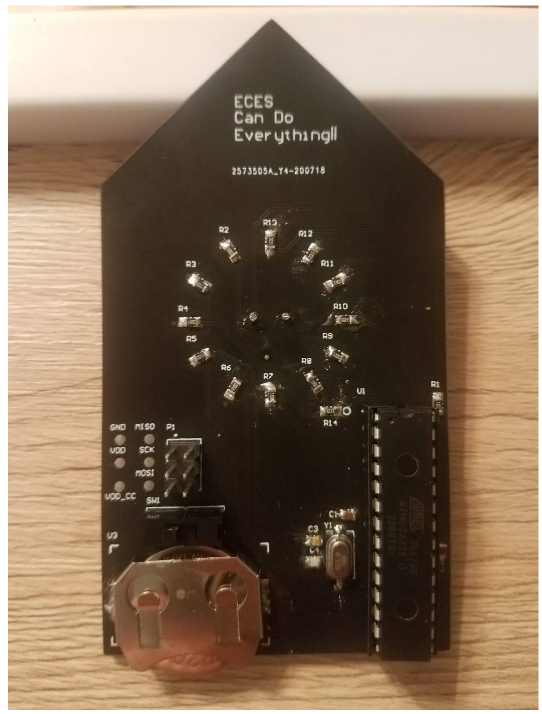
Clock Tower Badge Design

The Clock Tower PCB design can be seen below. Its purpose is to play the Cornell alma mater on a piezo buzzer while flashing a series of LEDs in a clock formation. This project, as your first project, has been pre-specified for you, but we will go through the process of component selection and system design as an exercise.





Spec’ing out Your Design

To find links to these components, you can just search the MPN or check the BOM

1. Right off the bat, we need a piezo buzzer and LEDs to fulfill the PCB’s purpose. We chose the ASMB-KTF0-0A306 multicolor LEDs and the PS1240P02BT piezo buzzer.

2. What will give the LEDs and buzzer the sequence of inputs they require? We will need a microcontroller (MCU). For the purposes of this class we have decided to opt for the ATMEGA328p: this is the default MCU used on an Arduino UNO. It is easy to load on a bootloader and connect to the Arduino IDE, which is an easy beginners tool for programming these chips. For packaging, I have selected a DIP package because it is easier to hand solder, but other packages exist with the same functionality

    a. This MCU comes with an internal oscillator for a clock. However, it is only guaranteed for within 10% accuracy which is kind of garbage. So we will be using an external oscillator! These are calibrated for better accuracy.
    b. In order to easily test and program, we will use a DIP holder for easy removal of the MCU when desired.

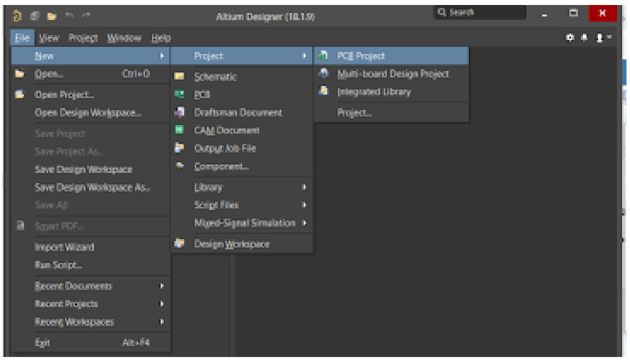
3. How will we power this system? It should be portable, so it needs a battery. The MCU we chose runs off 1.8-5.5 volts, so we chose the CR2032VP battery. Coin cell batteries require a holder to hold them onto the board and access the voltage, so we picked the appropriately sized battery holder, the 3034TR.

4. We don’t want this board running constantly, or the song would get annoying and the battery would die quickly. Therefore, we will add an SPDT switch to turn it off. We chose the EG1218 because it is a small, cheap rocker switch that can handle the small current required.

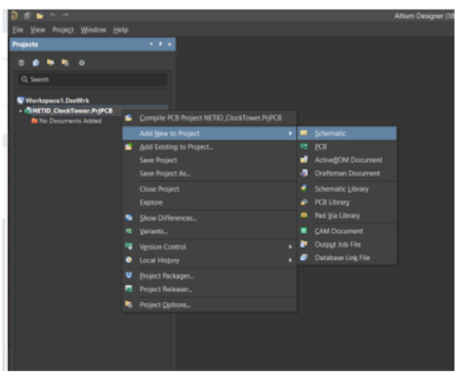
5. Some of these components will require resistors and capacitors (called, more generally, “passives”) to work properly, which we will get to in schematic. You will want to choose passives that can handle the power (not high in this case), and that are small but easy to solder. We will use 0805s in this case.

Getting Started in Altium

1. You should have already installed Altium. If not, go back to the Altium Set Up Guide
2. Open Altium. While you are doing so (this will take a while), go to your file system and create a folder called NETID_ClockTower
3. Start a new PCB Project: File -> new -> Project -> PCB Project. Right click to save the project as NETID_ClockTower



4. The first thing a project needs is a schematic! Right click on the project and click Add New to Project -> Schematic. Save it in the same way you saved your project, in the same folder.



5. There are several ways to manage components within projects and across projects. Especially if you are working at a company designing PCBs, they will likely have their own component management system. For now, you don't have a component management system, so you will have to obtain or design components for your projects. Altium has something called a Vault, where you can source components by searching their parts numbers. HOWEVER: you should be very cautious of this for more complex components, because they can often be wrong (incorrect pad dimensions, incorrect pin mapping, etc). For now, we will supply the majority of components to make this board and have students make a single component to get the concept of how to do so. Download the zip file Clock_Tower_NETID.zip, and we will make the last component later on.

You are now ready to start design!

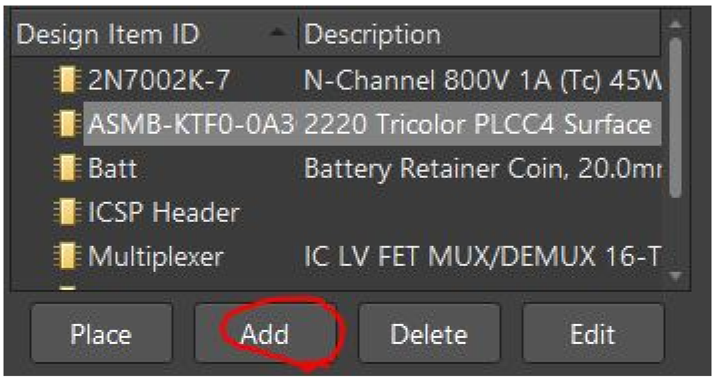
Creating a component

We will make the MCU in CAD ourselves, both the symbol and footprint, so that you know how. At some companies people called librarians make components, but if you are making boards on your own you will have to make many components in the future!
Part:https://www.digikey.com/product-detail/en/microchip-technology/ATMEGA328P-PN/ATMEGA328P-PN-ND/2357094

Please watch the posted video to see a component being created!

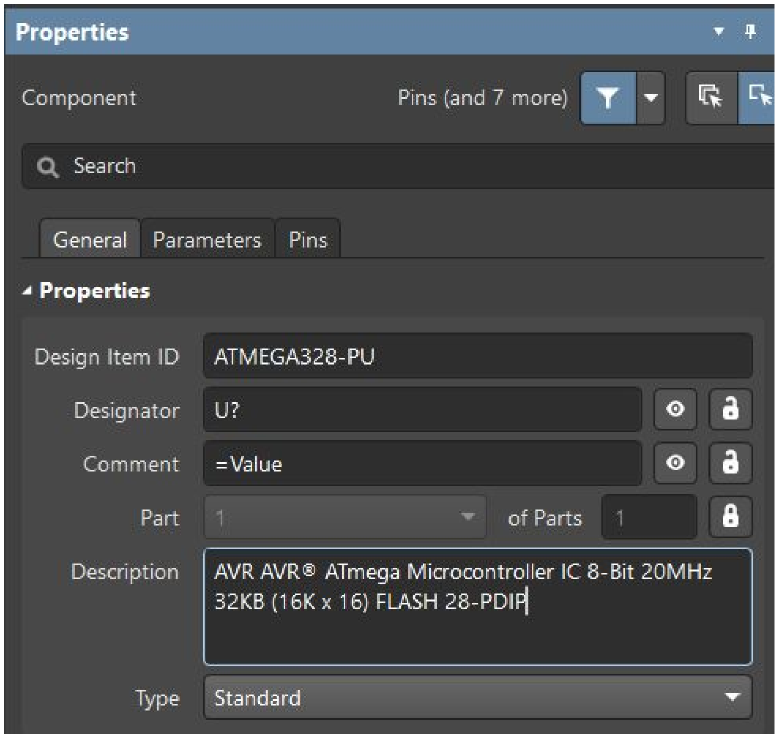
Schematic Symbol

1. In the provided schematic library, create a new component (simply click "Add").

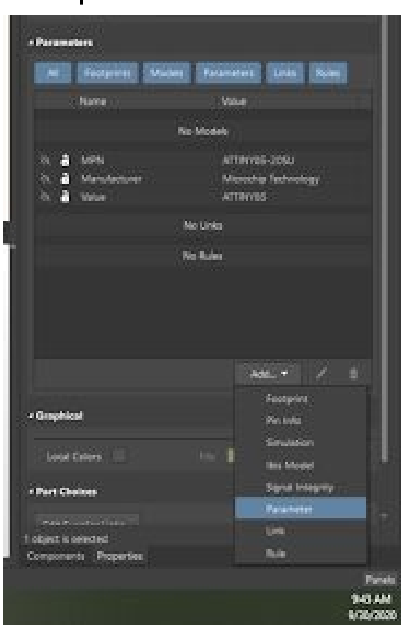
   

   In the box that comes up, you should name the component how you want it to be named on your schematic. Generally for an MCU, you will want to call this by the name of the chip, here ATMEGA328P.

2. In the Properties Panel to the right of the screen, there will be two tabs; General, and Pins. Click on General, where there will be categories for Design Item ID, which we have already assigned, Designator, Comment, and Description.
   a. Designator is the type of component, of which there are many, including resistors, capacitors, headers, crystals, and ICs. An MCU is an IC, which has the designator U. We will assign U?, which will allow Altium to auto-number it for us later. Here you can find a list of every type of designator, as defined by IEEE https://en.wikipedia.org/wiki/Reference_designator
   b. Comment should be =Value, which will be assigned later in Parameters. This will allow it to show up on the schematic itself
   c. For description, you can put whatever you'd like, but generally I default to at least the info in the description section of Digikey
   d. When you are done, the General section of the Properties panel should look like such



3. Next we will scroll down to the Parameters tab. These components will be loaded onto the BOM, so you will want to ensure you or someone you are working with has enough info to order the parts based on the values here. To add parameters, click Add->parameters



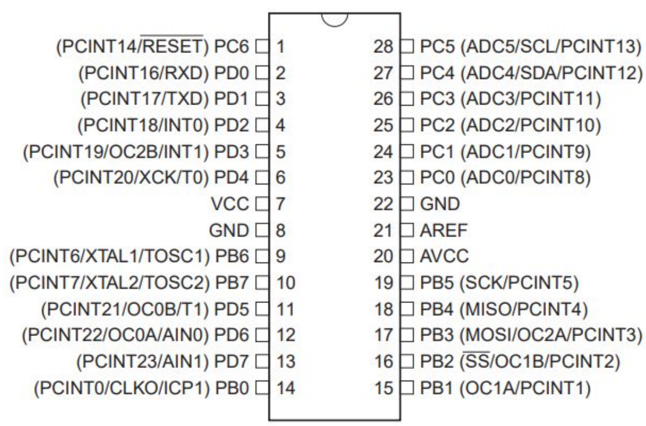
   a. The first parameter should be Value. This is what will show up on the schematic, as we set Comment = Value in General. The string here depends on the type of component; generally for passives you will put the value of the passive, like 10K for a resistor or 1uF for a capacitor. For an MCU, I would generally put the shorted part number, here ATMEGA328p

b. Next, we should add a manufacturer. This is listed on the DigiKey listing

c. The final crucial parameter is MPN (Manufacturer Part Number). This is how you will find your part from your manufacturer. This is listed on the DigiKey listing. Ensure you use the Manufacturer part number and not the Digikey part number: Digikey is the supplier, not the manufacturer.

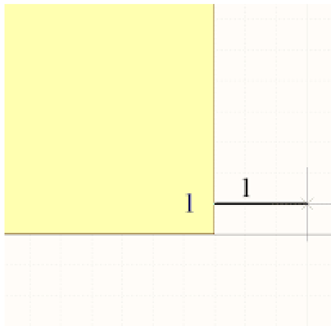
d. There are many, many other helpful parameters you could add, such as package descriptions, temperature ratings, datasheet links, material, lifecycle, current/voltage ratings, etc.

4. Pull up the datasheet for the component https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf
5. Near the top will be the pinout information. We are using the 328P package, which has 28 pins. Be sure to copy the correct pin diagram for the 28 pin dip



a. Note that when making a schematic component, as long as pins are numbered correctly, they do not need to be in the correct order. This is NOT true for the footprint. Therefore, you will find in schematic that it is easier for routing to place power pins at the top of the page, and ground at the bottom. More on that later.
b. Now onto actually making the component. In order to place anything on the symbol, you can use the shortcut “p”, which will open a menu of options for placing. The first thing we should place is the outline, which is “pr” (place rectangle). Drag and click to place the rectangle down. Don’t worry about the exact size right now, it is very easy to adjust.

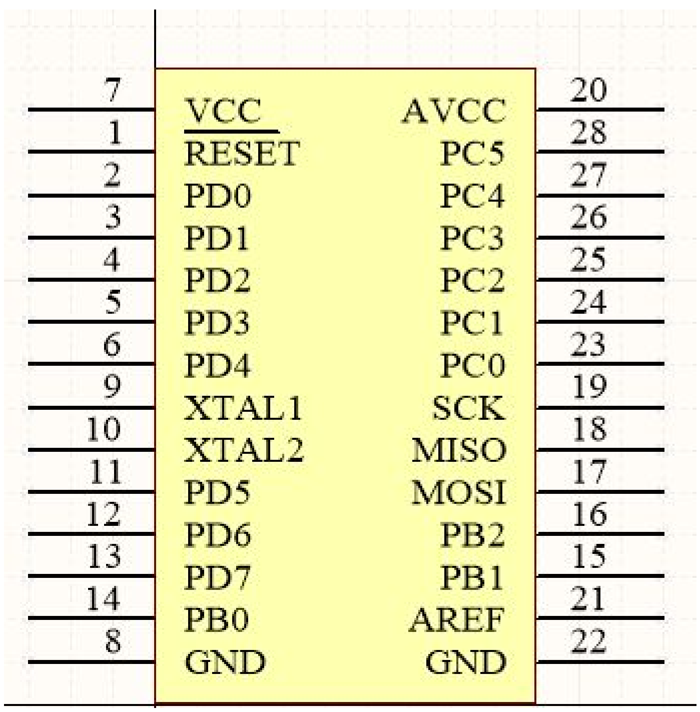
c. Now we will place all the pins. The shortcut for this is "pp". The pin should be placed such that the name is inside the block, and the pin number is outside



d. You can copy and paste on both sides to get 28 total pins. To put pins on the other side of the block, press the spacebar twice to rotate or press x to flip horizontally.

e. When you double click on a pin, it will print up the Properties menu again. The pin designator should be the number, and the name should be the description "VCC", "GND", etc. Generally you should put all the pin info, but for this case we can just use the description of what we will be using. For now, label your pins so that the component looks like below

    i. The Reset pin has a bar over it indicating that it is active low, or if you pull it low it will reset the chip. To name it like this, type R\E\S\E\T



f. Done with symbol! We will come back to this library to add the footprint after we have made it.

Footprint

Now we will make the footprint, which is the representation of the physical component. An important concept to review is layers; onions have layers, PCBs have layers. In a two layer board, you have top and bottom layers, which are the copper layers of the top and bottom of the board. For the purposes of footprint, you also have silkscreen, which is the text/drawings that show up on the board. There are the mechanical top and bottom layers, which are where the physical components go. You have a courtyard, which is a mechanical layer, which is just a keep-out area around the component to remind yourself not to put components too close together. Don't worry about the rest for now.

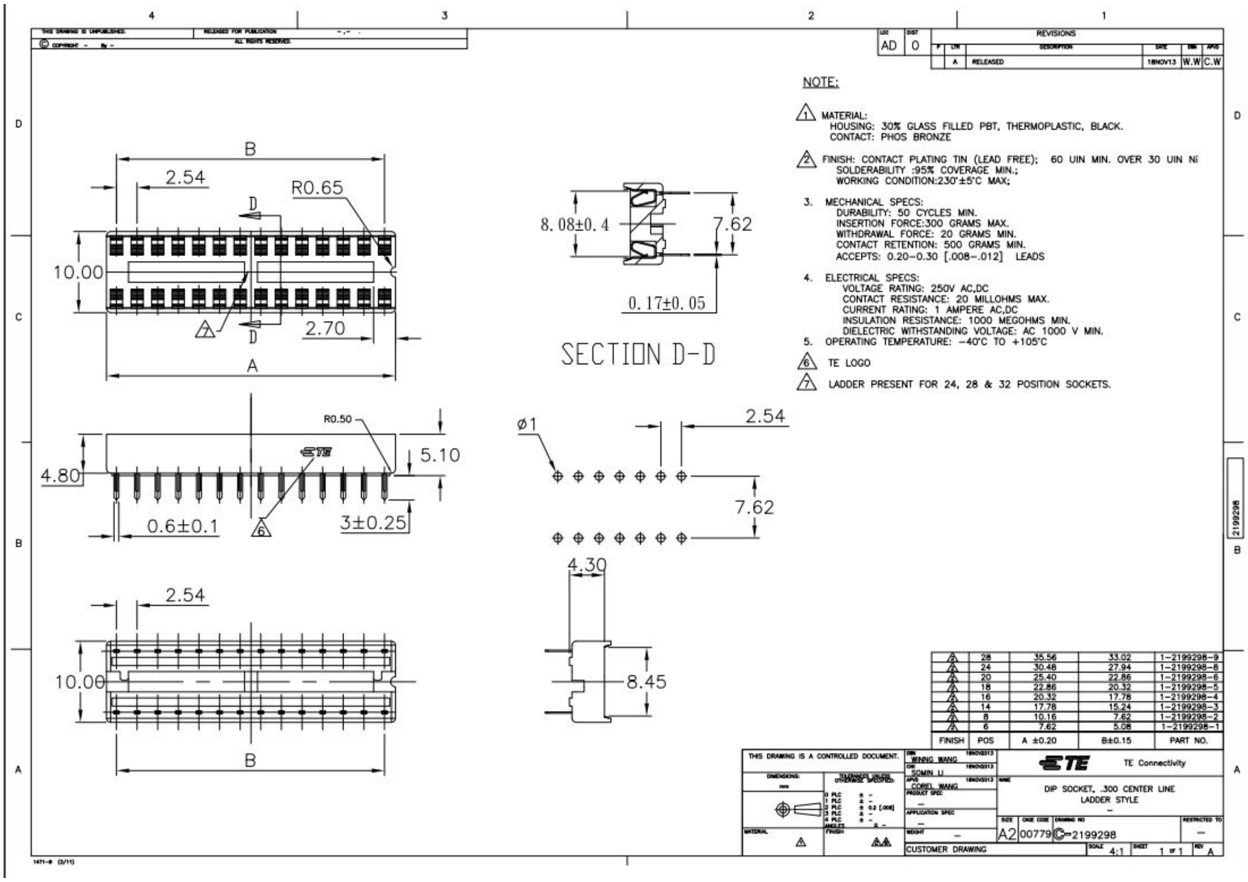
Top/Bottom Metal, Mechanical(M), Top/Bottom Overlay (silkscreen)

In many cases for making standard ICs, you can use the IPC Compliant Footprint Wizard, which will auto-generate a component given its dimensions.

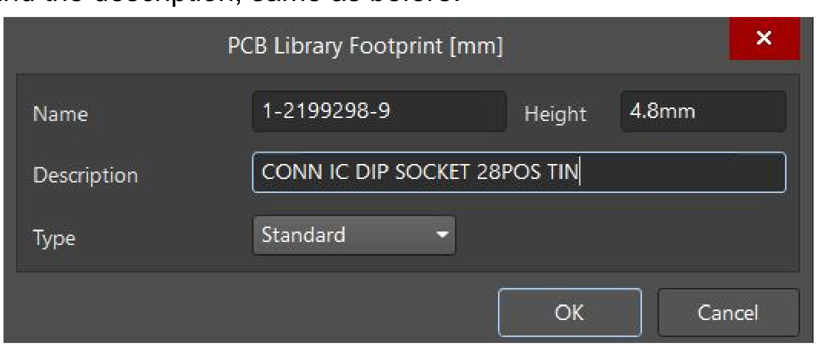
When designing with a DIP package, you can either directly design the DIP itself, or you can use something called a DIP socket. This means you can pop the IC in and out of its holder, which is nice if you accidentally break or brick it. We will do this because they are kind of easy to break.

You need to select a DIP socket with the same pin count and pin spacing as the DIP package you will be placing into it. This is the one Arduinos use here:
www.digikey.com/products/en?keywords=A120353-ND
Open up the drawing for mechanical sizing

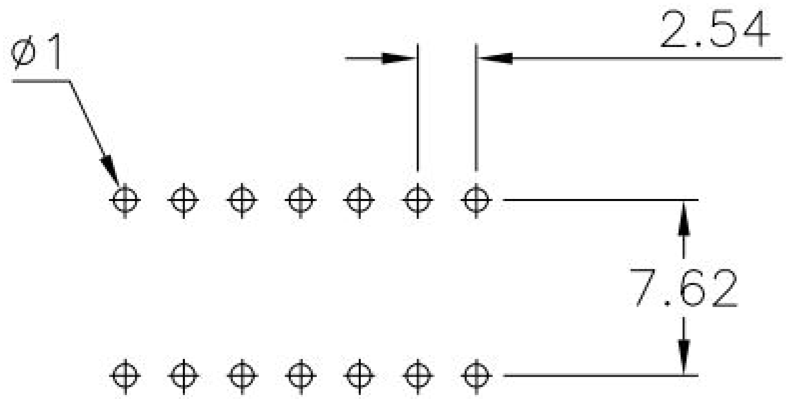
The Manual Way:



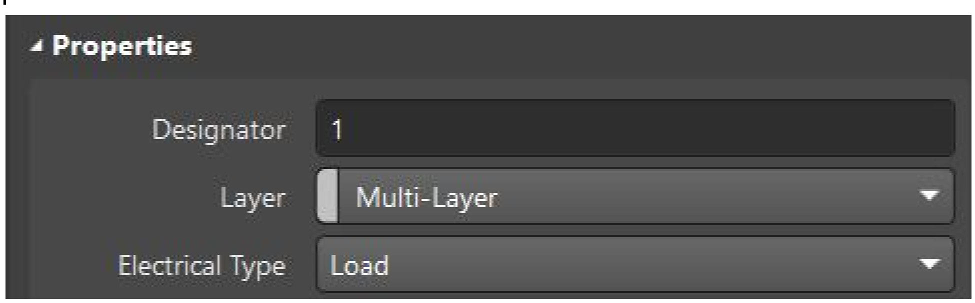
1. Open up the pcbLib document provided in the zip, and add a new component just as you did for schematic. Double click on the component to name it with the value of the component and the description, same as before.



2. First we will place pads. Using the same shortcut to place pins, press “pp” and place the first pad at the origin. The drawing helpfully gives layout of the pins themselves

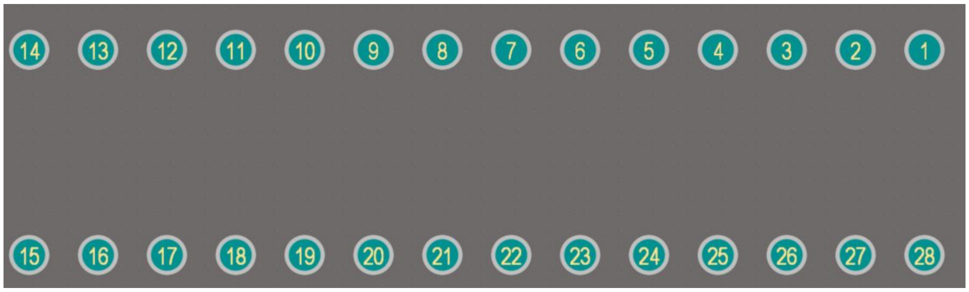


   a. We want to give a bit of tolerance on the pins to make sure they can fit into the pads, so we will make the holes slightly bigger than listed here.
   b. Again using the properties menu, make sure the designator is 1 and the Layer is multi-layer, as this will be a plated through hole (PTH) part. This means the leads will go all the way through the board, and they are plated with metal to provide contact to the rest of the board.

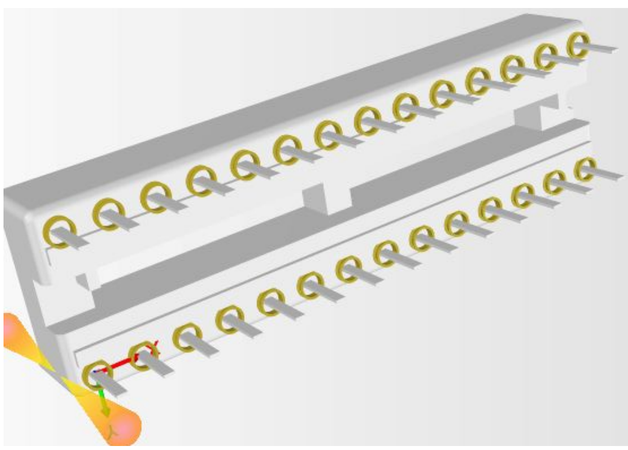


   c. Scroll down to Hole Information, and make the hole size 1.2 mm (which is a 20% tolerance). There are different ways to tolerance for actual manufacturing, but this will do the trick.
   d. Scrolling down to size and shape, make sure the shape is round. The size here indicates the amount of copper that will surround the hole, which should just be enough that you can solder to it. 1.5mmx1.5mm should be adequate.
   e. Now that you have this pad, you can copy and paste it and space them out as shown in the above diagram, 2.54 mm apart in the row and two rows spaced 7.62mm apart. This is arduous.

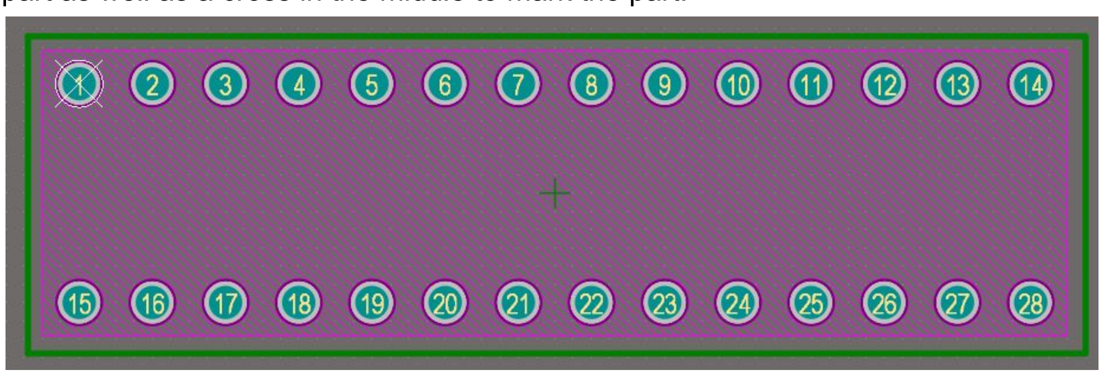
f. Your pads should now look like this



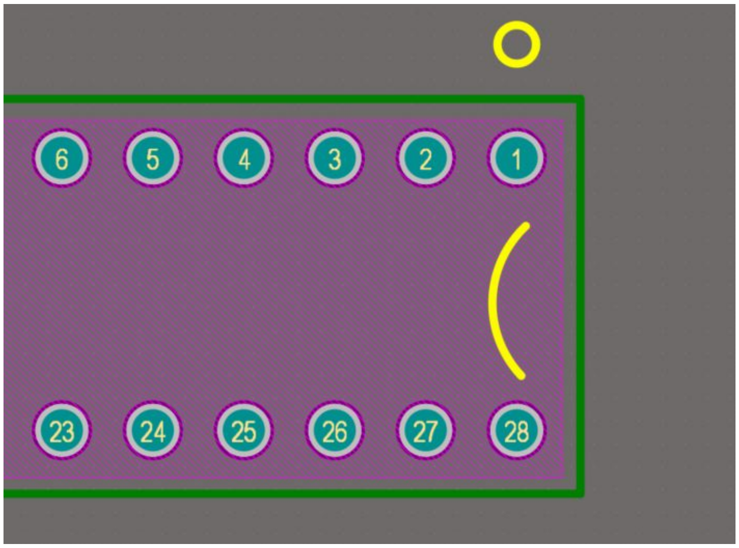
3. Now we will place the 3D body. You can download this from Digikey under the Documents and Media section.
    a. To place the 3D body, click "PO" -> 3D body.
    b. Navigate to where you have saved the 3D body and select.
    c. To better line up the 3D body and the pads, click 3 on your keyboard to go into 3D view. You will see that the body is below the pads: to remedy this, in the 3D Model Type window, you can select a Standoff Height. Use shift+right click to rotate the part, and ensure the pads are lined up with the legs. This is a good time to ensure you spaced the pads properly; if you did not, the part will not line up.



    d. Click 2 to go back to 2D mode.
4. Now that the body is placed, we can place a courtyard. Select M15 from the menu of layers at the bottom of the screen and click PL to place a line, creating a box around the part as well as a cross in the middle to mark the part.



5. As you probably noticed, this part is symmetric. DIP packages have a mark on the side facing pin 1, so you can mark the silkscreen the same way to remember to put it in the right way. Make sure it will be visible after the DIP socket has been placed.



6. There are many other markings you can add to your part, but this will suffice for this part. You are done with footprint! Now, we just need to go back to the schematic library to link the parts. Be sure to save the library.
    a. When back in the schematic library, click on the ATMEGA part again, and in the editor menu, select Add Footprint.
    b. Click Browse and select the pcbLib as Library, and you should find the matching footprint you just created. Click OK, OK, and then save your library. The part is now done! Good job :)

The Wizard Way

1. Go to Tools -> Footprint Wizard
2. Open up the data sheet for the ATMEGA328P linked above
3. Select DIP as the type
4. Click next, and copy over the parameters from the datasheet
5. Click “finish”
6. Link the schematic symbol to the footprint in the schematic library as described above
7. That’s it!