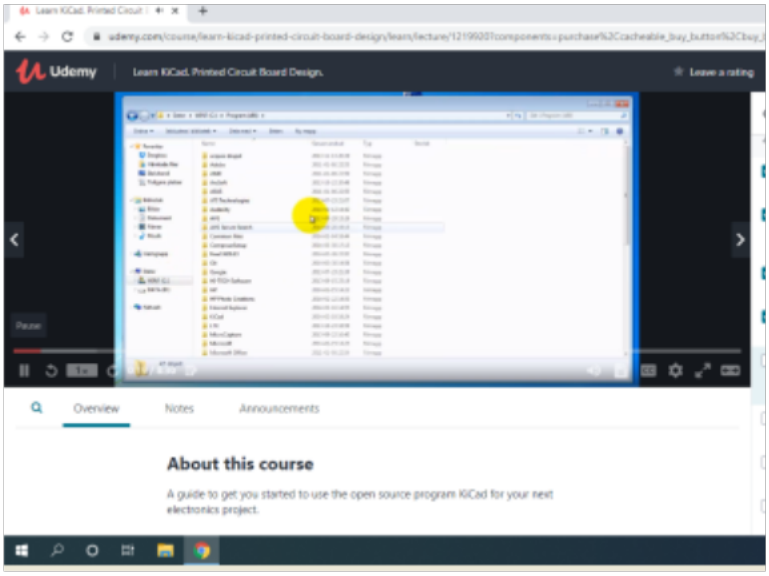
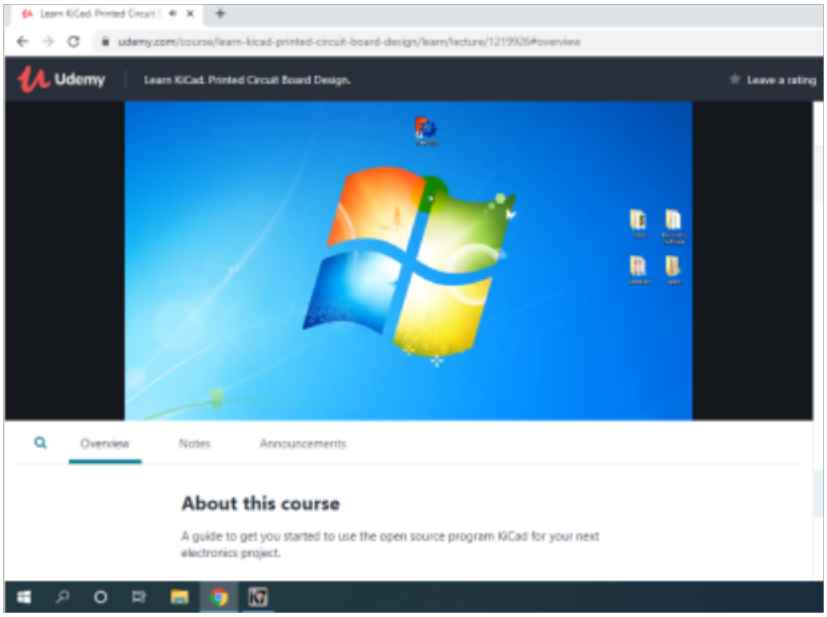
Date:	10-06-2020	Name:	POOJA K S
Course:	PCB	USN:	4AL17EC070
Topic:	Silk screen and copper pour.maintaining holes.create a library and put your own component in that library.	Semester and section:	6th sem and 'B' sec
Repository name:	pooja-shivanna		





Mounting holes:

Mounting holes are on every PCB design, but there is very little documentation about this subject matter. A Google or Wikipedia search on “Mounting Holes” renders no solutions to the PCB designer. Another issue that interferes with standardization is Imperial Unit ASNI hardware and ISO Metric hardware. So we’re going to have to explain both unit systems for clarity. But first let’s start with the basic fundamentals that both unit systems have in common.

The supported mounting hole usually gets tied to the GND plane without a Thermal Relief (a direct connection is best) and the supported hole w/vias gets both the main hole and the vias tied to the GND plane. Due to the fact that mounting hardware never gets soldered to the PCB, there is no reason for a Thermal Relief pattern and you connect all holes (including vias) directly to the plane. The unsupported (non-plated) hole has no connection to a GND plane layer and they require an outer layer keep-out defined that compensates for the hardware tolerances. See figure 2 for an illustration of the slop tolerance of a flat washer and the necessary copper keep-out sizing. There are two primary reasons for adding vias to the supported mounting hole. The first was to insure that if the screw threads stripped the copper plating from the main hole that the vias would still provide adequate ground connections. The second reason was for additional support to prevent the PCB from crushing when too much torque was used to tighten the nut. The average via hole size for mounting holes is 0.5 mm. See Figure 3 for a supported mounting hole with vias.

Create a library:

The library name should be unique within the chosen library table. It should also communicate what footpirnts to expect within it to your target audience. Do not include special chars inside your library name as it might create problems in some platforms.
This means the most shareable name will only contain letters, numbers, underline and minus. (This is only a suggestion. Add additional chars at your own risk.After creating the library you will need to add the library to the global or project library table. Adding it to the global table will make this lib visible to all your projects. The project (also known as local) library table only adds it to the current project.

Create a PCB footprint:

Altium Designer hosts a huge array of ready-made PCB Components both in servers as well as in several integrated and discrete libraries available through AltiumLive. However, even with this rich set of resources, it is likely that at some

point in your career you will need to create a custom PCB Component. PCB Component Footprints are created in the PCB Library editor using the same set of primitive objects available in the PCB editor. In addition to footprints, company logos, fabrication definitions, and other objects required during board design can also be saved as PCB Components.The real-world component that gets mounted on the board is represented as a schematic symbol during design capture, and as a PCB footprint for board design. Altium Designer components can be:Created in and placed from local libraries or Placed directly from a managed content server, which is a globally accessible component storage system that contains thousands of components, each with a symbol, footprint, component parameters, and links to suppliers.

Steps to Create your Component Footprint:

Creating your footprint in Altium consists of 4 steps:

1.Create the pads
2.Define component height and area
3.Add silk screen information
4.Save the footprint

Let's step through the process to see how easy it can be to create your component footprint.

Here's how to create your footprint in Altium Designer in 4 easy steps:

Step 1: Create the Pads

You will need the landing pattern for your part, which can be found towards the end of the component data sheet. For this example, let's use the popular PIC24FJ64GA004 microcontroller. This component is packaged in a 44-lead plastic thin quad flatpack.In Altium Designer, under File → New → Library → PCB Library. This will add a new PCB footprint library to your project. You'll also need to add new components to your PCB Library file. When you create a new PCB Library file, the library will create a blank footprint (named PCBCOMPONENT_1) by default.

Step 2: Define Component Height and Area

In this step, we need to define the height and area occupied by the component. We also need to define the component type. To access this information, select your new component footprint from the Footprints list, and click the Edit button. From here, you'll be able to enter these three pieces of information.By default, the

component type will be set to Standard; this is the value we would want for this component. Other components, such as mechanical elements and no-BOM components, will not be standard components and should be assigned the appropriate component type

Step 3: Add Silk Screen Information

For this step, we add the silk screen layer image and pin 1 marking. We will follow the suggestion from the data sheet and indicate only where the corners should be. To make a corner, we create a 0.08 mm line which you get by selecting the line icon on the PCB Lib Placement toolbar, duplicate it (by copy and paste) and link them. Here, make sure the silk screen information is assigned to the correct layer. Here, we want to place this on the Top Overlay layer. This can be done by selecting the silk screen lines from the Properties panel.

Step 4: Save the Footprint

The final step is to create your component is to name and save it so you can add it to your component , which also includes the schematic symbol. Tip: You will want to make the name unique and searchable so you can easily locate it.

Date:	10-06-2020	Name:	POOJA K S
Course:	JAVA	USN:	4AL17EC070
Topic:	Getting user input.do while.arrays.	Semester and section:	6th sem and 'B' sec

- Getting User Input :-

```
import java.util.Scanner;
public class App{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a floating point value:");
        double value = input.nextDouble();
        System.out.println("You entered :" + value);
    }
}
```

- Do While :-

```
Scanner scanner = new Scanner(System.in);
int value = 0;
do{
    System.out.println("Enter a number:");
    value = scanner.nextInt();
}
while(value != 5);
System.out.println("Got 5!");
}
```

- Switch :-

```
import java.util.Scanner;
public class Application {
    public static void main (String[] args) {
        Scanner input = new Scanner (System.in);
        System.out.println ("Please enter a command :");
        String text = input.nextLine ();
        switch (text) {
        case "start" :
            System.out.println ("Machine started");
            break;
        case "stop" :
            System.out.println ("Machine stopped.");
            break;
        default :
            System.out.println ("Command not recognized");
        }
    }
}
```

- Arrays :-

```
int value = 7;
int[] values;
values = new int[3];
System.out.println (values[0]);
values[0] = 10;
values[1] = 20;
values[2] = 30;
System.out.println (values[0]);
System.out.println (values[1]);
System.out.println (values[2]);
```

```
for(int i=0; i<values.length; i++){
    System.out.println(values[i]);
}
int[] numbers = {5, 6, 7};
for(int i=0; i<numbers.length; i++){
    System.out.println(numbers[i]);
}
}
}
```

- Arrays of Strings :—

```
Public class App{
    public static void main(String[] args){
        String[] words = new String[3];
        words[0] = "Hello";
        words[1] = "to";
        words[2] = "you";
        System.out.println(words[2]);
        String[] fruits = {"apple", "banana", "pear", "Kiwi"};
        for(String fruit : fruits){
            System.out.println(fruit);
        }
        int value = 0;
        String text = null;
        System.out.println(text);
        String[] texts = new String[2];
        System.out.println(texts[0]);
        texts[0] = "one";
    }
}
```