Lab 5: Digital to Analog Converter (DAC)

ECE 302, 302H

Check out:

- Multimeter with probes

Gather in lab:

- 2 kΩ Resistor x 5
- 1 kΩ Resistor x 3
- Wire Kit
- Protoboard
- Wire Stripper
- Solder Spool
- MSPM0 Microcontroller

Motivation:

It is often important in electrical engineering to convert digital representations to analog outputs. Digital to Analog Converters (DACs) are tools that take digital bits as inputs and output a corresponding voltage. In this lab, we will explore a simple design (R-2R DAC) to generate analog voltages from a four-bit input. We will utilize a microcontroller to output a digital value, and the circuit you will design will map that value to a set of discrete analog voltages.

Soldering

Soldering is an important part of bringing a circuit design to production. After a PCB has been designed, components can be attached to the PCB using solder. Solder is a metal alloy that has a relatively low melting point, which makes it a perfect candidate to fuse metals together. In the context of PCBs, solder is used to connect the leads of components to the pads on the PCB.

Component packages come in a variety of forms. In ECE 302, you will interact with two main categories of packaging: through hole technology (THT) and surface mount devices (SMD). The soldering technique for different types of components varies, but most of the principles are the same. This document will walk you through soldering a through hole resistor.

The first step in soldering a device is to place the device on the PCB, such that you can solder without the component moving. For a through hole resistor, the best way to do this is to bend the leads after inserting it into the PCB. Figure 1 gives an example of how this looks like.

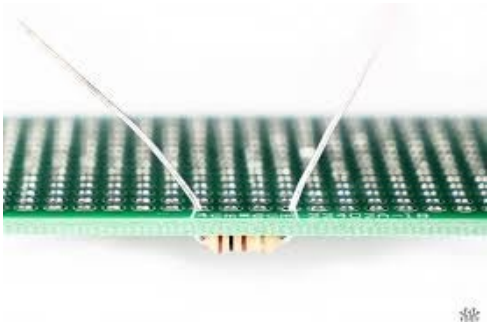

Figure 1 – Unsoldered resistor in PCB

Once the component is secured in place, it is ready to be soldered. This requires the use of a soldering iron and the solder. Additionally, you can use tools, such as flux to make the process easier.

Flux is used to make the solder flow better and create a stronger joint. It also helps clean up oxidization on pads and leads to make the fused connection more secure.

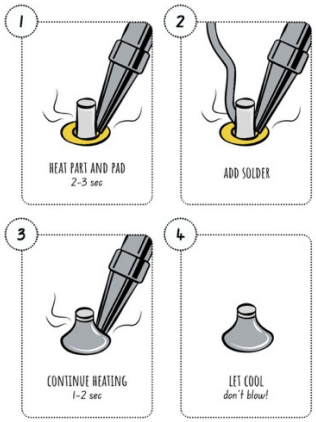

Figure 2 – Steps for a good solder; Source – Circuitmess.com

To solder, turn on the iron to the desired temperature. Most lead-free solder requires temperatures of 650 to 750 degrees Fahrenheit. Once the iron reaches the required heat, place the tip of the iron on the pad. Then, hold the solder to the pad. The heat from the pad should melt the solder and help it fuse the lead of the component to the pad on the PCB. To make sure the joint is secure, make sure the solder fuses both the pad and lead together. In many cases, if the pad isn't heated enough, the solder won't create a strong connection. Refer to figure 2 for a visual description of creating a good solder joint.

After a couple of attempts, you're solder joints should start to look like the perfect joint in figure 3. Finally, you can use snippers to cut the excess leads off the components. This will prevent accidental shorts on your circuit. The result will look like the perfect joint in figure 3.

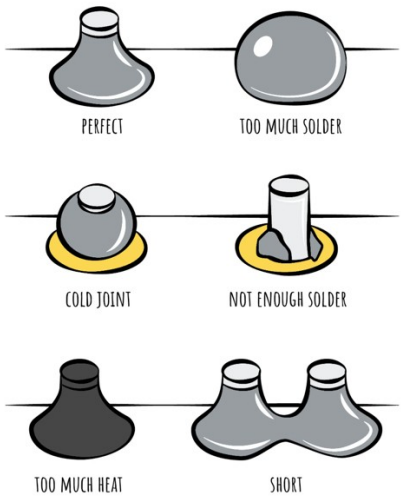

Figure 3 – Different conditions of solder; Source – Circuitmess.com

Warning: Make sure you "tin" your soldering irons after you use them or before taking a break. This means you should coat the tip of the soldering iron with fresh solder and then turn them off. This prevents the iron from oxidizing and keeps the equipment in our labs functioning well.

Custom Protoboard

Similar to the breadboard found in lab 1, protoboards are a more permanent option for prototyping designs. Protoboards are generally a PCB with several evenly spaced pads, which can be soldered together to create flexible circuit designs. Several protoboards provide built-in connections like breadboards. However, in other cases, engineers must solder wires or create solder bridges to create connections between components. The figure below shows an example of the custom protoboard you will use in this class.

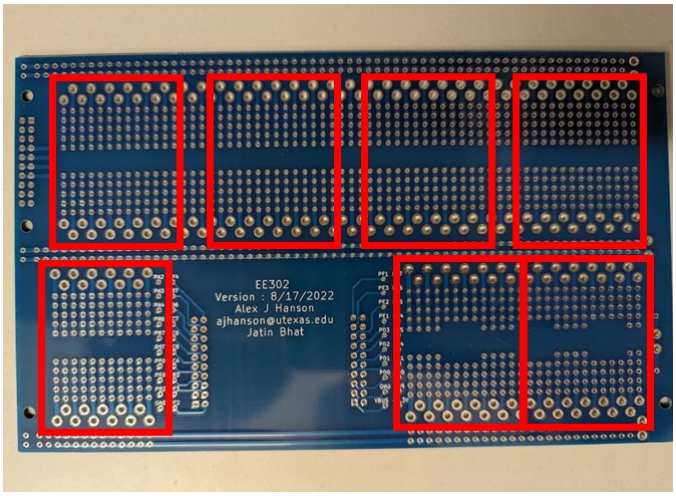

Figure 4 – Proto board divisions for circuit construction.

The board is divided into 7 regions to reduce wasted PCBs. The different lab sections will share the same protoboards. Ensure that your design is contained within a single box from figure 4. This custom protoboard has a division in the middle of its rows and it contains power rails like the breadboards. The PCB is routed to mimic the internal metal clips in breadboards. If you want to connect two nodes together, you can use a jumper wire to do so. Make sure to solder all components to the board to ensure proper contact.

Code Composer Studio

Code Composer Studio is the development environment you can use for the microcontroller we use in this class. Install code composer studio (CCS) via the latest download on the website: www.ti.com/tool/CCSTUDIO. Navigate to downloads and select the appropriate file for your operating system. This documentation covers the usage for version 20.3.0 on Windows, but most of the concepts should apply to other contexts as well.

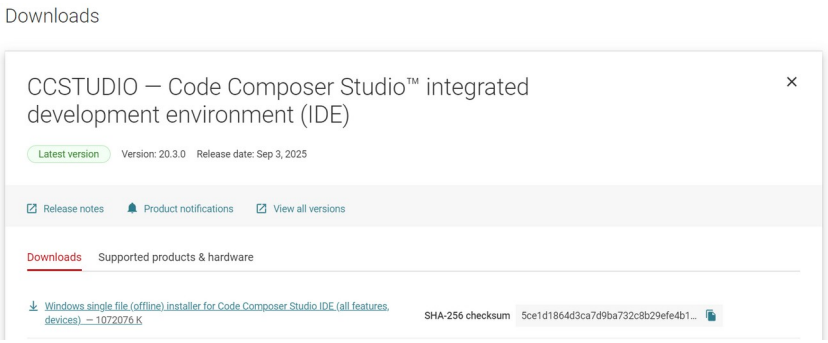

Figure 5 – Download page with the Windows link shown

Unzip the downloaded file and start the "ccs_setup_<version_number>.exe" file found inside. When the installer asks you to select which components you want to install, choose the same options as figure 6.

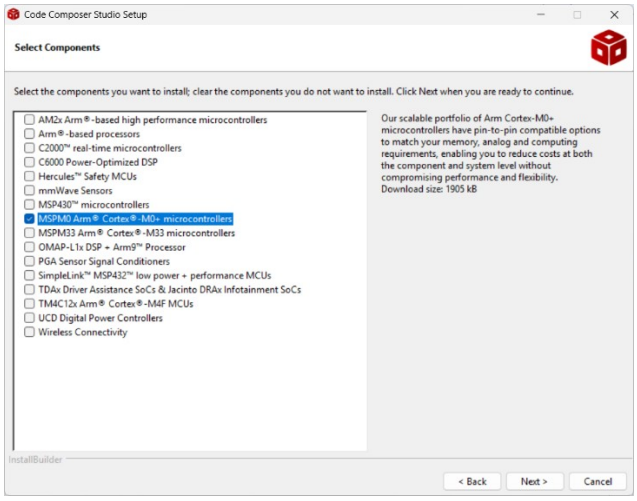

Figure 6 – Select Components Window

Continue with the installation as normal. Once the installation is complete, open CCS and import your project. Use the option shown in figure 7 to browse the folder containing your project.

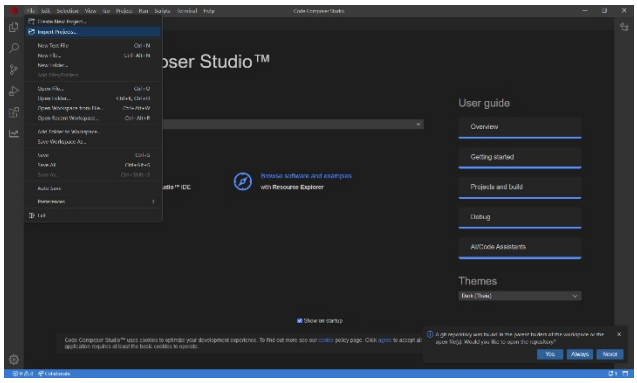

Figure 7 – Import project menu

Once the import menu is opened, browse for your folder and select the options found in figure 8 to continue.

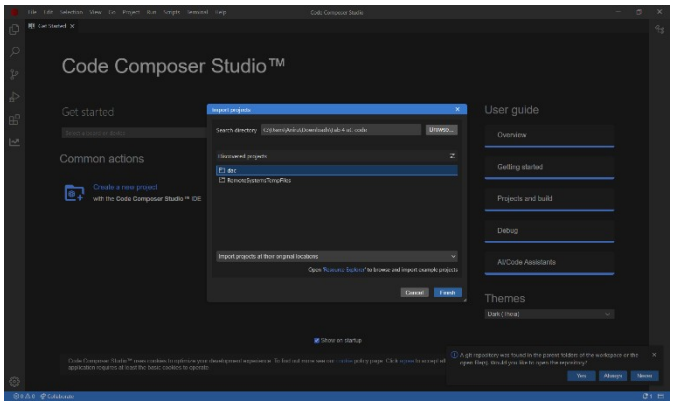

Figure 8 – Import projects from assigned location

Once the project is open, you can open specific files from the folder menu on the left. You may have to select the file icon on the top left to view this menu. When you're ready to run your script, make sure your microcontroller is connected to your laptop. Open the debug interface

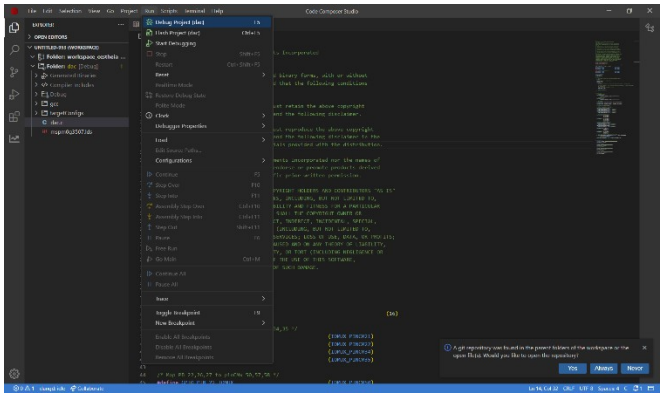

Figure 9 – Debug project menu

The debug page should resemble figure 10. You will see several options to interact with your microcontroller. This allows for live feedback to the operation of your script.

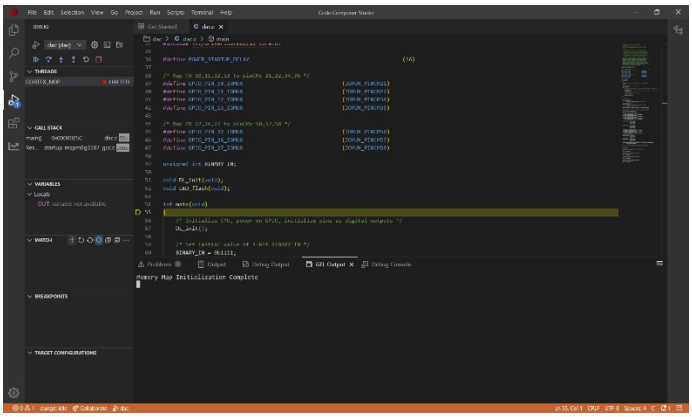

Figure 10 – Debug menu with add watch highlighted

Before you start your script, add any variables you want to watch. This can be done by clicking the plus icon while hovering over the “Watch” box on the left debug menu. Type in the variable name into the popup window and add it to watch list.

Finally, you can run the script by selecting the run button highlighted in figure 11.

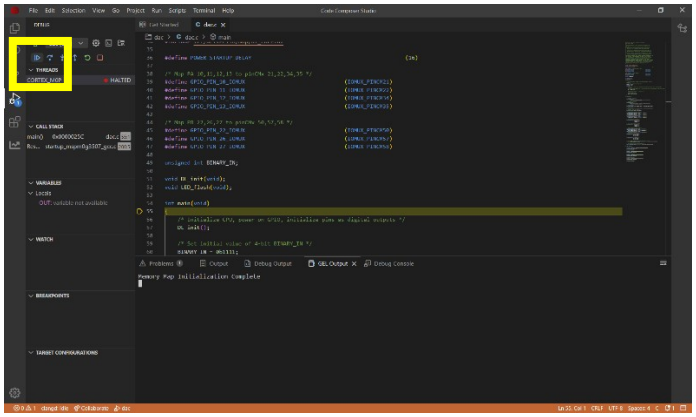

Figure 11 – Run icon hovered over on top left

While the script is running, you can update variables by right clicking on the variable name in the watch list. Select “Update Expression Value” and set the variable accordingly. The microcontroller should react to the change immediately.

Microcontroller: MSPM0G3507

The MSPM0 is LaunchPad is a development board for an ARM based processor. You will learn more about the specific microcontroller in future courses. The purpose of the microcontroller in this lab is to provide controllable voltage sources. These sources correspond to pins on the development board. You can turn them on and off through your computer using code composer studio: the IDE for Texas Instruments microcontrollers.

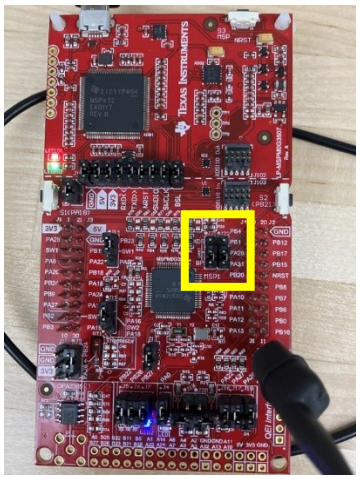

Figure 12 – Microcontroller Launchpad with required jumper configuration

The microcontroller has several pins (male adapters on the front and female adapters on the back). The ground node is exposed as one of these pins. The other pins are also labeled and output a high (3.3V) or low (0V) output for this class. To ensure that the code in this lab functions correctly, make sure the jumpers are in the configuration highlighted. Both jumpers should be placed on the bottom two pins.

Instructions:

Part 1: You will build a 4-bit R-2R digital to analog converter on a proto-board. The circuit can be found in figure 1. The voltage sources in the schematic represent the outputs of pins on your microcontroller. The sources will be confined to output either High or Low.

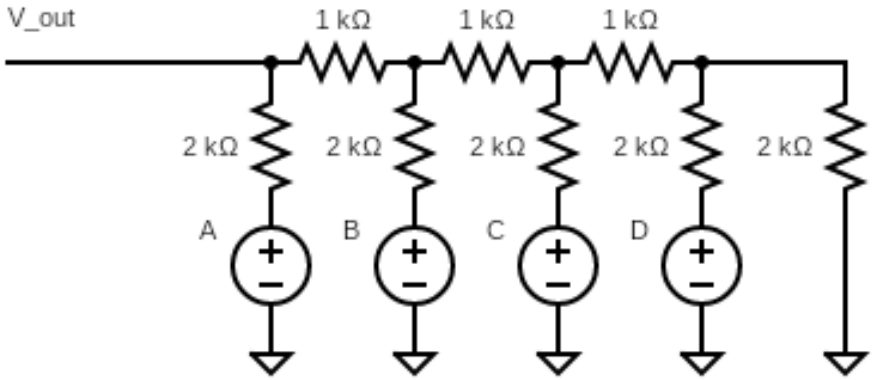

Figure 1 – Schematic of a 4-bit R-2R DAC

1. Place the resistors on a proto board. To ensure that other lab sections can use the same proto boards, make sure to keep your circuits to within one red box in figure 2.

Hint: The proto board is set up in a very similar way to the breadboards you have already worked with. There is a power rail to assist with setting up the circuit as well.

2. Next, solder the resistors on the proto board after you have verified their placement.
3. Solder any wires to connect resistor nodes if needed
4. Solder wires onto your protoboard for each input voltage source.

Hint: You will want to attach a couple of wires to ground so you have easy clipping spots for any probes during measurement.

Warning: Make sure you "tin" your soldering irons after you use them or before taking a break. This means you should coat the tip of the soldering iron with fresh solder and then turn them off. This prevents the iron from oxidizing and keeps the equipment in our labs functioning well.

Part 2: Powering the DAC

1. We will not need a power supply for this lab, since the "power" will come from the microcontroller which will be connected by USB to your laptop.
2. Ensure that the GND node of your circuit is tied to the GND pin of the micro controller. This does the same thing as connecting the negative terminal of the voltage sources to GND in the schematic.
3. Connect the micro-controller (uC) pins to the voltage source nodes in your proto board. Use the pin mapping found below.

DAC Net	uC pin
D	PA10
C	PA11
B	PA12
A	PA13
GND	GND

Part 3: Setting up the microcontroller

1. Connect your microcontroller launch pad to your computer using USB
2. Run Code Composer Studio (CCS); instructions for installation can be found in the lab documentation above.
3. Import the project “dac” which can be downloaded from Canvas into CCS.
4. Add “BINARY_IN” the watch variables list.
4. Start the script

Warning: Make sure that the microcontroller is configured as described in the documentation above. Any inaccuracy in the jumper clips on the microcontroller can lead to the script not working as intended.

Task #1: Enter the decimal value of BINARY_IN from the table at the end of the document in the watchlist. Measure V_out in your circuit using a multimeter and record it in the chart at the end of the lab. Also fill out the expected voltage based on your calculations from the prelab.

Lab Submission:

A	B	C	D	Decimal Equiv.	Expected V_out [V]	V_out [V]
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1	**15**	**3.094**	