UNIVERSITY OF CALIFORNIA, BERKELEY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES

EE 113B/213B: Power Electronics Design
Module 0: Component Sizing, Soldering, and Microcontroller Setup

Objectives

By the end of this module, you should be able to...

- Size converter components based on design specifications
- Solder through-hole and surface mount components
- Test solder connections using an multimeter
- Remove soldered components and clean PCBs
- Open, edit, and compile firmware projects in Texas Instruments Code Composer Studio (CCS)
- Produce PWM signals using the Texas Instruments C28x Microcontroller
- Use the CCS debugger to change and read variables and debug code
- Use an oscilloscope to measure generated waveforms

Recommended reading: KPVS Sections 5.1-5.3, 5.6-5.7, 22.4, 25.1-25.2

Pre-lab Assignment Shikai Shen.

Welcome to EE 113B/213B! Throughout the semester, we will design, prototype, and validate a maximum power point tracking (MPPT) converter for a PV module. To get started, let's assume we plan to use a synchronous buck topology, shown in Fig. 1. Notice the PV module is modeled as a current source. In this first pre-lab, we will size the converter's components. We will also download and install the Code Composer Studio (CCS) integrated development environment (IDE) as well as the C2000Ware software development kit (SDK).

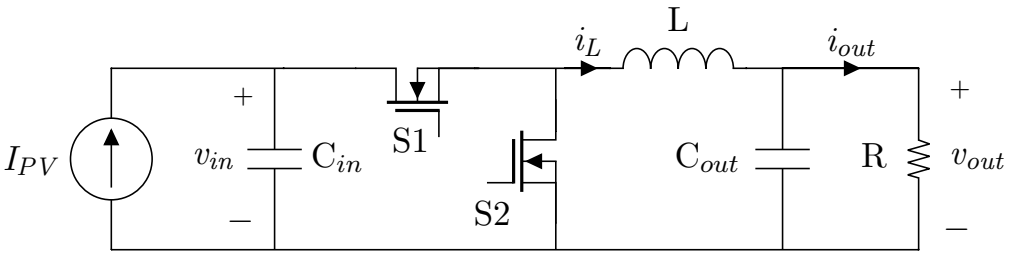


Figure 1: Synchronous buck converter.

Table 1: Buck Converter Design Specifications

Specification	Value
Nominal Input Voltage	20 V
Input Voltage Range	16-24 V
Output Voltage	12 V
Nominal Output Power	100 W
Output Power	50-100 W
Switching Frequency	100 kHz
Input Voltage Ripple	$\leq$ 5 %
Output Voltage Ripple	$\leq$ 5 %
Inductor Current Ripple	$\leq$ 20 %

Size Components

Assume your buck converter has the specifications shown in Table 1. For all of the following calculations, you may assume deadtime is negligible. Hint: You are encouraged to write a script for these sizing calculations in case you change your design later.

1. Calculate the duty cycles required for the nominal operating point and all four corner operating points (i.e., all four combinations of highest load, lowest load, highest input voltage, and lowest input voltage).
2. Assuming a peak-to-peak inductor current ripple of no more than 20% at full power, calculate the minimum inductance required for L.
3. Assuming a peak-to-peak output voltage ripple of no more than 5% for any operating point, calculate the minimum capacitance required for C_{out}.
4. Assuming a peak-to-peak input voltage ripple of no more than 5% for any operating point, calculate the minimum capacitance required for C_{in}.
5. Calculate the maximum voltage and current stress for each switch (i.e., each power transistor).
6. How would choosing a higher switching frequency affect your answers for questions 1-5? Explain.

Welcome to EE 113B/213B! Throughout the semester, we will design, prototype, and validate a maximum power point tracking (MPPT) converter for a PV module. To get started, let's assume we plan to use a synchronous buck topology, shown in Fig. 1. Notice the PV module is modeled as a current source. In this first pre-lab, we will size the converter's components. We will also download and install the Code Composer Studio (CCS) integrated development environment (IDE) as well as the C2000Ware software development kit (SDK).

photovoltaic

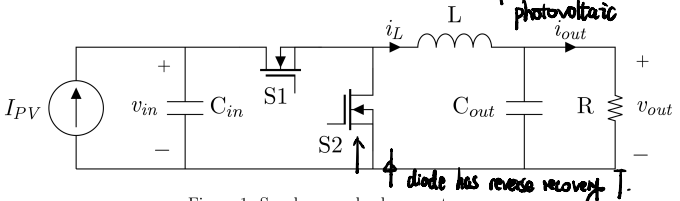


Figure 1: Synchronous buck converter.

diode has reverse recovery T. × high freq.

Table 1: Buck Converter Design Specifications

Specification	Value
Nominal Input Voltage	20 V
Input Voltage Range	16-24 V
Output Voltage	12 V
Nominal Output Power	100 W
Output Power	50-100 W
Switching Frequency	100 kHz
Input Voltage Ripple	≤ 5 %
Output Voltage Ripple	≤ 5 %
Inductor Current Ripple	≤ 20 %

load

$\frac{V_o - V_i}{L} \cdot DT_s$

$\frac{10-5}{36\mu H} \times 0.5 \times 10^5 = 694mA$

① ⟺

$T_s = \frac{1}{f_s} = 10^{-5}s$

Size Components

Assume your buck converter has the specifications shown in Table 1. For all of the following calculations, you may assume deadtime is negligible. Hint: You are encouraged to write a script for these sizing calculations in case you change your design later.

1. Calculate the duty cycles required for the nominal operating point and all four corner operating points (i.e., all four combinations of highest load, lowest load, highest input voltage, and lowest input voltage).
2. Assuming a peak-to-peak inductor current ripple of no more than 20% at full power, calculate the minimum inductance required for L.

24V 16V

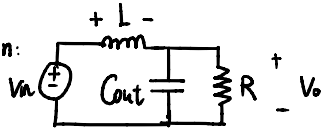
For Buck Converter, $V_{out} = D \cdot V_{in}$.

1. Nominal operating point:

$V_{in} = 20V$, $P_{out} = 100W$, $V_{out} = 12V$, $D = 0.6$

2. $\frac{di_L}{dt} = \frac{V_L}{L}$

S_1 on:

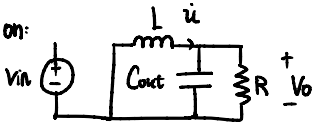


1.2 highest V_{in}, highest load ⟺ P_{max}

⇒ $V_{in} = 24V$, $P_{max} = 100W$, $V_{out} = 12V$.

⇒ $D = 0.5$

S_2 on:



$\langle V_L \rangle = D \cdot (V_{in} - V_o) + D' \cdot (-V_o) = 0$

$\langle i_L \rangle = \cdots = 0$

⇒ $\langle i_L \rangle = I_L = \frac{V_o}{R}$.

1.3 highest V_{in}, lowest load ⟺ P_{min}

⇒ $V_{in} = 24V$, $P_{min} = 50W$, $V_{out} = 12V$.

⇒ $D = 0.5$

$2\Delta i_L = \Delta i_{LPP} = \left| \frac{V_L}{L} \cdot DT_s \right| = \left| \frac{V_o - V_i}{L} \cdot DT_s \right|$

$\Delta i_{LPP} \le i_L \cdot 20\%$, $\Rightarrow \left| \frac{V_o - V_{in}}{L} \cdot DT_s \right| \le \frac{V_o}{R} \cdot 20\%$

$\Rightarrow L \ge \frac{5(V_{in} - V_o) \cdot DT_s}{V_o} \cdot R$, $R = \frac{V_o^2}{P}$

$\Rightarrow L \ge \frac{5(V_{in} - V_o) \cdot DT_s \cdot V_o}{P}$

1.4 lowest V_{in}, highest load ⟺ P_{max}

⇒ $V_{in} = 16V$, $P_{max} = 100W$, $V_{out} = 12V$

⇒ $D = 0.75$

max

when $D = 0.5$, $L \ge \frac{5(24-12) \times 0.5 \times 10^{-5} \times 12}{50} = 3.6 \times 10^{-5} H$

1.5 lowest V_{in}, lowest load ⟺ P_{min}

⇒ $V_{in} = 16V$, $P_{min} = 50W$, $V_{out} = 12V$, $D = 0.75$

3. Assuming a peak-to-peak output voltage ripple of no more than 5% for any operating point, calculate the minimum capacitance required for C_{out}. $2\Delta V_c \le 5\%$

4. Assuming a peak-to-peak input voltage ripple of no more than 5% for any operating point, calculate the minimum capacitance required for C_{in}.

5. Calculate the maximum voltage and current stress for each switch (i.e., each power transistor).

6. How would choosing a higher switching frequency affect your answers for questions 1-5? Explain.

3. $\frac{dV_c}{dt} = \frac{i_c}{C}$, $I_c = I_L - \frac{V_o}{R}$, $\Rightarrow \Delta i_c = \Delta i_L$ (neglect V_c).

$q = C \cdot 2\Delta V_c = \frac{1}{2} \cdot \frac{1}{2} T_s \cdot \Delta i_L$

$\Rightarrow \Delta V_c = \frac{T_s \cdot \Delta i_L}{8C}$, $\left(\Delta i_L = \left|\frac{V_o - V_{in}}{2L} \cdot DT_s\right| = \right)$, $2\Delta i_L = 20\% I_L \Rightarrow$ $2\Delta i_L = 20\% \times \frac{100}{12}$ or $20\% \times \frac{50}{12}$

$= \frac{5}{3}$ or $\frac{5}{6}$

$\Rightarrow 2\Delta V_c \le V_c \cdot 5\% = V_o \cdot 5\% = 12 \times 5\% = 0.6V$

$\Rightarrow \frac{T_s \cdot \Delta i_L}{8C} \le 0.3V$

$\Rightarrow C \ge \frac{10^{-5} \times \frac{5}{6}}{8 \times 0.3} = 3.47 \mu F$ Cout

---

4. $\frac{dV_c}{dt} = \frac{i_c}{C}$, from KCL for small ripples: * $\Delta i_c = D \cdot \Delta i_L$ (on-state)

$\Rightarrow \Delta V_c = \frac{I_c}{C} \cdot T_s$, $\Delta V_c \le \frac{5\%}{2} \times V_c = 0.3V$

$\Rightarrow$ To simplify calculation

$\frac{I_c}{C} \cdot T_s \le 0.3$

$\Rightarrow C \ge \frac{D \cdot \Delta i_L \cdot T_s}{0.3} = \frac{0.75 \times \frac{5}{3} \times 10^{-5}}{0.3} \approx 4.17 \times 10^{-5} F$ Cin

---

DTs & δTs

$I_c = I_{pv}$ or $I_L - I_{pv}$

$= \frac{P_{out}}{V_{in}}$ or $\frac{P_{out}}{V_{out}}$

choose ---

5. Q1: $V_{stress,Q_1} = V_{in} + \Delta V_{in} = (1+2.5\%) \times V_{in} = 24.6V$

$I_{stress,Q_1} = i_L + \Delta i_L = (1+10\%) \times I_L = 9.17A$

Q2: $V_{stress,Q_2} = 24.6V$

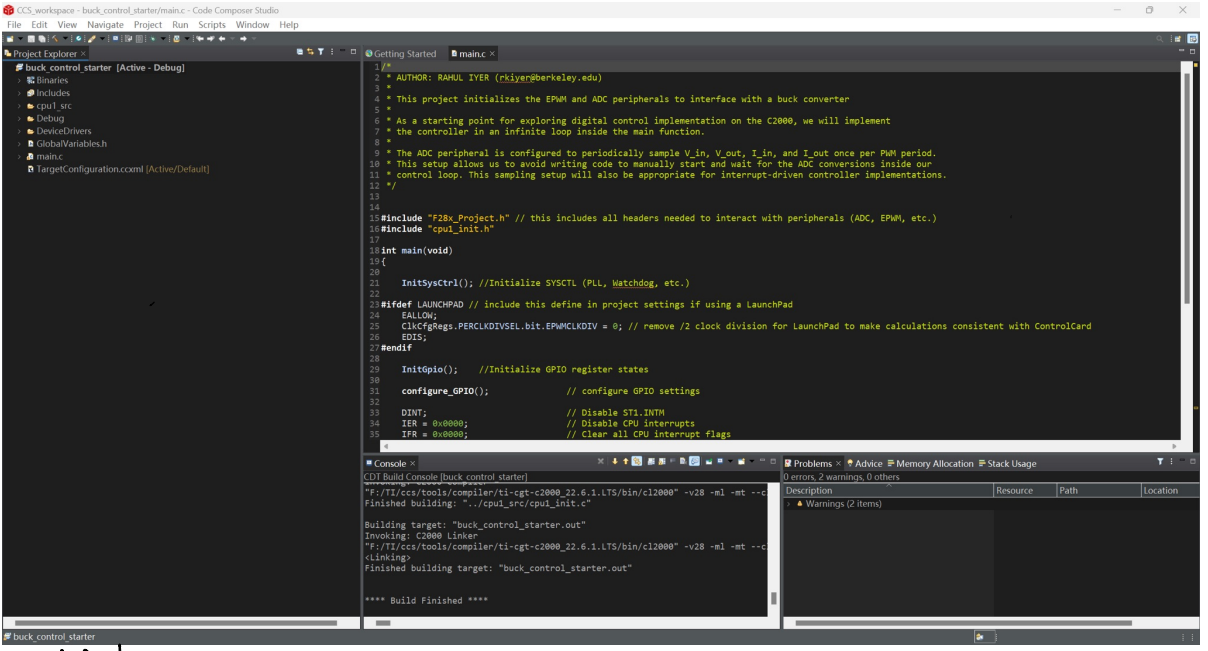
since P=50W for worst-case

$I_{stress,Q_2} = 9.17A \times 0.5 = 4.59A$

x2

6. when f ↑, T_s ↓, ⇒ current/voltage ripples ↓,

To maintain the same ripples, smaller L, C are better choices.



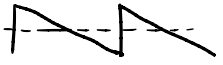
build finish.

Digital PWM Calculations

Now calculate the following, referencing the EPWM section of the technical reference manual as needed. You may assume $f_{\text{EPWMCLK}} = 100\,\text{MHz}$ and $f_{\text{sw}} = 100\,\text{kHz}$.

1. Number of EPWMCLK counts per switching period
2. For a leading-edge carrier, calculate the value of TBPRD. Explain your reasoning.
3. For a trailing-edge carrier, calculate the value of TBPRD.
4. For a triangular carrier, calculate the value of TBPRD. Explain your reasoning. (**Hint: the calculation for a triangular carrier is different from that for a sawtooth carrier**).

1. counts per switching period = $\frac{f_{\text{EPWMCLK}}}{f_{\text{sw}}} = \frac{100\text{MHz}}{100\text{kHz}} = 1000$

2. leading-edge: 

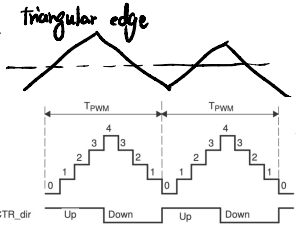
counts = 1000 = TBPRD+1

$\Rightarrow$ TBPRD = 999

3. trailing edge: 

$\Rightarrow$ TBPRD = 999

4. triangular edge



TBPRD is actually achieved by counter and register,

2TBPRD = 1000

$\Rightarrow$ TBPRD = 500

5. In **GlobalVariables.h** update the value of the preprocessor definition `#define EPWM_TBPRD` with the value you calculated in step 4 (we are using a triangular carrier)

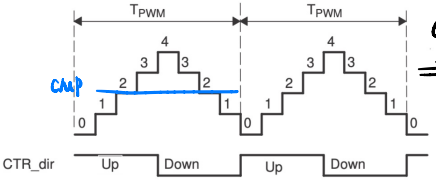
6. We want to initialize our code to start PWMs with 50% duty ratio. Update the value of `#define EPWM_CMP_INIT` accordingly (We are using a triangular carrier).

7. Draw the carrier, modulating waveform, and resulting duty switching signal corresponding to our initial duty ratio.

8. Assume we want 50 nanoseconds of deadtime. Calculate the number of EPWMCLK ticks that elapse in 50 nanoseconds. Update the value of `#define EPWM_DEADTIME` with the value you calculated.

5. ✓

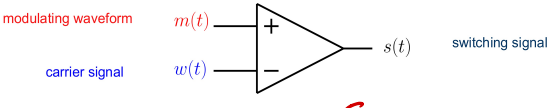
6.

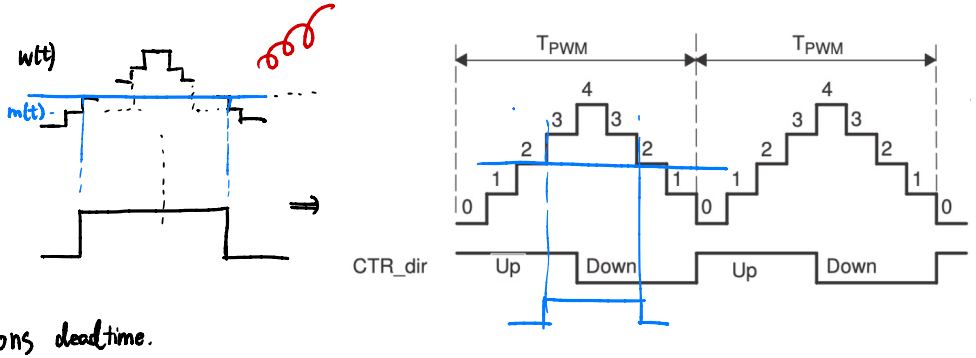


CMP+1+CMP-1 = $\frac{1000}{2}$

⇒ CMP = 250

7.





8. 50ns deadtime.

$$EPWM\_DEADTIME = \frac{50ns}{t_{EPWMCLK}} = \frac{50ns}{\frac{1}{100MHz}} = 5$$

Getting Started | main.c | GlobalVariables.h

```
#ifndef GLOBALVARIABLES_H_
#define GLOBALVARIABLES_H_

#define EPWM_TBPRD 499 500

#define EPWM_CMP_INIT 249 250     // initialize with 50% duty ratio
#define EPWM_DEADTIME 5                    // deadtime in EPWMCLK ticks

#define ADC_ACQ_WINDOW 30                  // ADC S&H Window Length in SYSCLK cycles


#endif /* GLOBALVARIABLES_H_ */

```

9. Maybe ≈ 3.5 hours - 4 hours (include some review of EE113).

Code Composer Studio Integrated Development Environment

Next, we will download and install the Code Composer Studio (CCS) integrated development environment (IDE) as well as the C2000Ware software development kit (SDK).

An Integrated Development Environment (IDE) is a software application that you can use to develop software. Code Composer Studio (CCS) is an IDE from Texas Instruments for developing software intended for Texas Instruments microcontrollers. Because the software we develop is "low-level" and interfaces directly with the hardware internal to the microcontroller, we typically refer to microcontroller software as *firmware*.

The following steps will guide you through the download and setup of CCS:

1. Download the CCS zip file for your OS (Windows or Mac) from bCourses. Note: If you are downloading CCS from elsewhere, you should ensure you select version 12.
2. Unzip the downloaded file if necessary and run the CCS installer.
3. Specify the custom installation option, and select *C2000 Real-Time MCUs*.
4. Continue clicking *Next* and wait until the installation finishes. The installation should take approximately 5-10 minutes. The CCS installer may ask you to reboot your PC to complete the installation.

   Note: on MacOS, you may be required to approve CCS in the system privacy and security settings. If the installer or application will not open, navigate to `System Settings > Privacy & Security` and perform the necessary approval actions.

C2000Ware Software Development Kit

A Software Development Kit (SDK) is a collection of software library files and tools that assists in developing software for a particular application. The C2000Ware SDK from Texas Instruments contains libraries, compiler tools, and examples that enable us to efficiently develop code for Texas Instruments C28x microcontrollers.

The following steps will guide you through download and setup of C2000Ware. **Note: you must wait until the CCS installation is complete before installing C2000Ware.**

1. Navigate to https://www.ti.com/tool/C2000WARE.
2. Click on *Downloads*.
3. Next to *C2000WARE*, click *Download options*.
4. Click on the appropriate installer for your computer's operating system.
5. Follow the steps to create an account and sign the software download agreement.
6. Unzip the downloaded file if necessary and install C2000WARE.

Note: on MacOS, the C2000Ware installer may be quarantined by Apple's security settings. Open the terminal and run the following command: `xattr -d com.apple.quarantine <path to C2000Ware installer>` replacing the argument in angle brackets with the path to the C2000Ware installer on your machine (eg: inside `\Downloads`)

Set Up Workspace and Import Project

To get you started, we are providing an example project that you will be modifying to control your converters. You will create a CCS workspace and import a copy of this example project into it.

1. Download and extract the *buck_control_starter* example project from bCourses.

2. In your file system explorer (Windows Explorer or Finder on Mac), create a new folder at a location of your choice, and title it with a name of your choice. This folder will serve to host the CCS workspace.
3. Run CCS. When prompted, select the folder you created in Step 2 as the workspace directory.
4. Click on File → Import.
5. Select Code Composer Studio → CCS Projects.
6. Next to *Select search-directory* click *Browse*. Navigate to and select the folder that you extracted in step 1.
7. Click the check-box next to the *buck_control_starter* project.
8. Click the *Copy projects into workspace* option.
9. Click *Finish* to complete the project import.

Now we will update the project to link to the version of C2000Ware we downloaded.

1. Right-click the *buck_control_starter* folder in the *Project Explorer* tab, and select *Properties*.
2. Under Resource → Linked Resources, click *$C2000WARE_ROOT$*. Click the *Edit* button on the right.
3. In the Location field, click *Folder*.
4. On Windows, navigate to `C:\ti\c2000` and select the folder starting with "C2000Ware". On Mac, navigate to `/Applications/ti/c2000` and select the folder starting with "C2000Ware".

The project should now build.

1. Right-click the *buck_control_starter* folder in the Project Explorer tab, and select *Build Project*. CCS will run the compilation steps internally. The build should complete after a few seconds with no errors and report "**** Build Finished ****".

Digital PWM Calculations

Now calculate the following, referencing the EPWM section of the technical reference manual as needed. You may assume $f_{\text{EPWMCLK}} = 100\,\text{MHz}$ and $f_{\text{sw}} = 100\,\text{kHz}$.

1. Number of EPWMCLK counts per switching period
2. For a leading-edge carrier, calculate the value of `TBPRD`. Explain your reasoning.
3. For a trailing-edge carrier, calculate the value of `TBPRD`.
4. For a triangular carrier, calculate the value of `TBPRD`. Explain your reasoning. (**Hint: the calculation for a triangular carrier is different from that for a sawtooth carrier**).
5. In **GlobalVariables.h** update the value of the preprocessor definition `#define EPWM_TBPRD` with the value you calculated in step 4 (we are using a triangular carrier)
6. We want to initialize our code to start PWMs with 50% duty ratio. Update the value of `#define EPWM_CMP_INIT` accordingly (We are using a triangular carrier).
7. Draw the carrier, modulating waveform, and resulting duty switching signal corresponding to our initial duty ratio.
8. Assume we want 50 nanoseconds of deadtime. Calculate the number of EPWMCLK ticks that elapse in 50 nanoseconds. Update the value of `#define EPWM_DEADTIME` with the value you calculated.

Assignment Feedback (Required)

How much dedicated time did you spend on this pre-lab?

Lab Assignment

Discrete power converters are most commonly prototyped using printed circuit board (PCBs). In the first half of this course, you will prototype your MPPT buck converter using a PCB we have pre-designed for you. Then, you will design your own PCB in the second half of the semester, creating your own power converter from scratch!

To get started, let's practice soldering electronic components to PCBs. We can broadly classify components as "through hole" or "surface mount" depending on how they are designed to attach to the PCB. This classification has implications for how we solder such components.

Note: Many types of solder contain Lead. You should always wash your hands after soldering, especially before eating or drinking, and beware of contaminating personal property.

Soldering Through Hole Components

Through hole components have leads that are inserted into plated through holes in a PCB and soldered into place. Before you begin, attach your PCB's standoffs and locate a soldering iron and strands of solder. Then, follow these steps to solder a banana jack connector to your PCB:

1. Begin heating the soldering iron to 320 °C (if your soldering iron has temperature control). Make sure the soldering sponge is moist; if not, add water.
2. Place the component on the PCB, inserting its leads into the appropriate through hole pads.
3. Secure the component in place by either taping it down or bending the leads where they stick out of the reverse side of the PCB.
4. In preparation for soldering, turn on the soldering fan and position its intake close to where you intend to work. Pick up the soldering iron and drag its tip through the sponge to remove excess debris. You should take these steps every time you begin to solder.
5. Once the soldering iron has finished pre-heating, you are ready to solder. Focusing on one lead, locate where it touches the through hole pad on the reverse side of the PCB. With one hand, hold the soldering iron so that its tip simultaneously heats the lead and the pad. With the other hand, hold the strand of solder so that its tip touches the pad, applying slight pressure to ensure contact with the pad. Note: We are using rosin-core solder, so additional flux is not necessary.
6. Once the end of the solder strand begins to melt onto the pad, the solder strand will shorten. Do NOT remove the soldering iron; continue to hold it in place. Re-position the solder strand so that its new end touches the pad, and allow it to melt again, adding more solder to the pad. Continue repeating this step until there is enough solder attaching the lead to the pad. Note: This step can also be accomplished in one long "push" of the solder strand into the pad once it begins melting the first time.
7. When you have finished adding solder, remove the solder strand. Allow the existing solder to reflow for a few more seconds. Then, remove the soldering iron. Your solder joint should look similar to the example shown in Fig. 2. **Note: Do NOT solder more than one of the connector's leads for now.** This will make removing the component in a subsequent section very difficult.

Checking Solder Connections

Once you have soldered a component onto a PCB, it is best practice to check its electrical connections with a multimeter as follows:

1. Turn on the multimeter and set it to measure resistance. Be sure the plug the test leads into the appropriate terminals for measuring resistance.

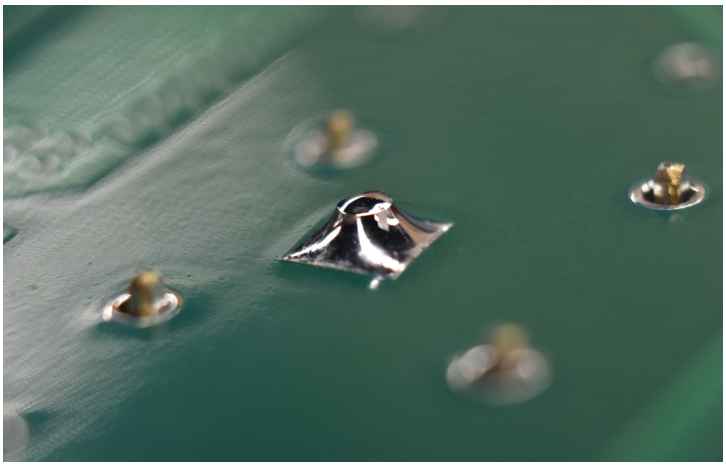


Figure 2: Soldered through hole component.

2. Use the probes to measure the resistance between an exposed section of the component's lead to somewhere else on the PCB that should be electrically connected to the lead. Do not probe the solder joint itself; the solder joint should be in the measurement path between your probes.

3. If the measured resistance is consistently close to zero, the solder connection is strong. If the the measured resistance bounces between near-zero and a high value, the solder joint is not reliably electrically connected. In the latter case, you should reflow the solder with the soldering iron (i.e., heat the pad long enough for the solder to re-liquify and resettle) and check the connection again once cooled.

Removing Soldered Components

Now, let's focus on removing soldered components. Remove the component from the PCB by doing the following:

1. Heat the soldering iron to the same temperature you used to originally solder the component.

2. Touch the soldering iron to the pad of the component you would like to remove.

3. Hold the iron in place until the solder has liquified and the component can be "nudged" out of place. This should require NO force. You should avoid trying to pull the component off; if you pull it off before solder has fully liquefied, you risk also damaging the PCB pad. This can make an entire PCB unusable!

4. Once the component can be "nudged" out of place, gently remove it using tweezers, keeping the soldering iron on the pad.

5. Lastly, remove the soldering iron.

Removing Solder from Pads

Once you have successfully removed your component without damaging any PCB pads (hopefully!), it is best practice to remove any remaining solder on the pad before re-soldering. This is commonly done using solder wick as follows:

1. Place the end of the solder wick strand directly over the solder you want to remove.

2. Place the soldering iron on top of the solder wick, directly above the solder you want to remove. Gently press down with the iron.

3. Hold the solder wick and iron in place until the solder liquefies and the solder wick "absorbs" it. Do NOT remove the iron at any point during this process; otherwise the solder will begin re-solidifying. If you need to stop solder absorption at any time, remove the solder wick and the iron at the same time.

4. Keeping the soldering iron gently pressed onto the solder wick, slide the solder wick around to other parts of the pad to absorb additional solder while the pad is hot. As the solder wick fills up, gradually move further up the strand (though don't waste unused solder wick).

5. Once all solder has been absorbed, remove the soldering iron and solder wick at the same time. Do not remove the soldering iron before the wick; if the solder re-solidifies, it may attach the wick to the PCB pad.

6. Trim and discard the used section of solder wick.

Now, practice the sequence of soldering the component onto the PCB, checking its connection, and removing it until you are comfortable doing each.

Check-off: Demonstrate (a) soldering one lead of the component, (b) checking the solder connection, and (c) removing the component.

Soldering and Removing Surface Mount Components

Surface mount components are commonly soldered using a hot air gun and solder paste. Solder paste contains small bits of solder suspended in flux paste, so use of additional flux is not needed.

Solder a surface mount capacitor onto your PCB as follows. You are encouraged to read all steps to make sure you understand them before beginning.

1. **Bring the air gun out to the front of your lab bench, and orient it so that its output is pointed away from equipment, computers, etc.** Begin heating the air gun to 380-400 °C. If there are other nearby components on your board made of plastic or other materials that may melt, create heat shields for them with foil to prevent them from melting. **Do NOT heat the air gun in its storage location at the back of your lab bench; this risks damaging the lab bench and other equipment, so the air gun should always be OFF when stored in this location.**

2. Place solder paste onto each of the capacitor's pads as shown in Fig. 3.

3. Using tweezers, place the capacitor onto the pads with its terminals touching the solder paste.

4. Once the air gun has fully heated, hold it in a vertical position pointing directly down at the capacitor, with the air output about 1-2 inches above the capacitor. Do NOT hold the air gun too close to the PCB; this can burn it. If the air gun blows the capacitor out of place, reduce its air flow and/or hold it higher.

5. Holding the air gun vertically, make small circles with the air gun around the capacitor to heat the PCB area evenly. This also helps to prevent burning the PCB. The solder paste will soon begin to liquify, turning a metallic silver color. If the solder paste does not liquify after awhile, consider increasing the air gun's flow rate.

6. Once the solder paste begins liquifying, do NOT remove the air gun. Continue holding it vertically over the capacitor and making small circles. With your other hand, you may use tweezers to nudge the capacitor into the correct position if needed. Removing the air gun at this point would likely require you to start over.

7. Once the solder paste has completely liquified and the capacitor is oriented in the correct position, remove the air gun.

8. Visually inspect the solder joints. Your solder joints should look similar to the example shown in Fig. 4. Check around the component for small solder balls and remove them with tweezers; solder balls risk shorting exposed pins and traces.

Now, let's practice removing that capacitor with the air gun.

1. Begin heating the air gun to the same temperature you used to solder the component, if it is not heated already.

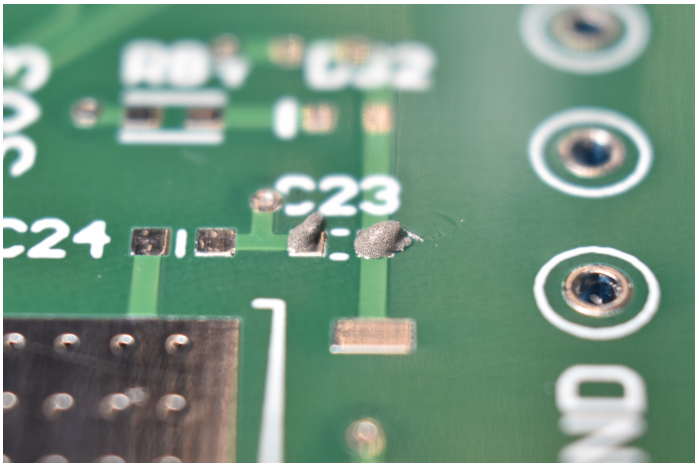


Figure 3: Solder paste applied for a surface mount capacitor.

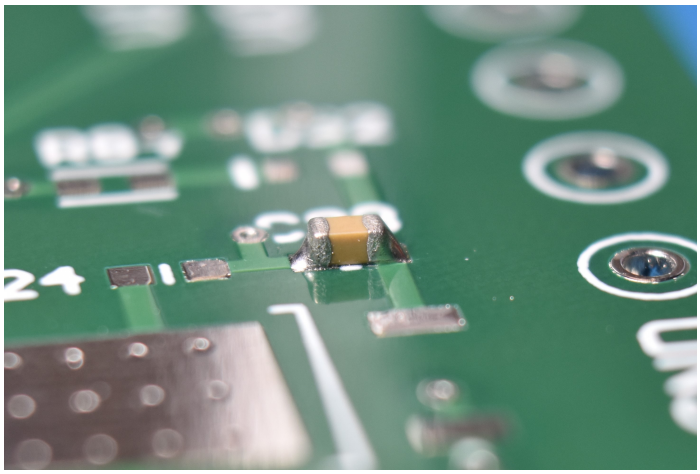


Figure 4: Soldered surface mount capacitor.

2. Once the air gun has fully heated, hold it in a vertical position pointing directly down at the capacitor, with the air output about 1-2 inches above the capacitor. Do NOT hold the air gun too close to the PCB; this can burn it.

3. Holding the air gun vertically, make small circles with the air gun around the capacitor to heat the PCB area evenly. This also helps to prevent burning the PCB.

4. Once the solder has liquified and the component can be "nudged" off, gently remove it using tweezers, still continuing to heat the area with the air gun. Removing the capacitor should require NO force. You should avoid trying to pull the component off; if you pull it off before solder has fully liquefied, you risk also damaging the PCB pad. This can make an entire PCB unusable!

5. Once the capacitor has been removed, remove the hot air gun.

Repeat these steps until you are comfortable soldering the capacitor onto the PCB using a hot air gun.

Soldering and Removing ICs

Hot air guns are particularly useful for soldering surface mount ICs and other components that have several pins, some of which may be on the component's underside. Now that you have practiced soldering a surface mount capacitor, practice soldering a SOIC-8 package op-amp IC as follows:

1. Begin heating the air gun to 320 °C, if it is not heated already. Note that certain ICs are more sensitive to temperature than others, so be sure to check manufacturer guidelines.

2. Place a small amount of solder paste onto each of the component's pads as shown in Fig. 5. For component packages with pins that are close together, too much solder paste often results in shorted pins, while too little paste often results in no connection. Try to dispense the same amount of solder paste onto each pad.

3. Using tweezers, place the component onto the pads with its terminals touching the solder paste.

4. Once the air gun has fully heated, hold it in a vertical position pointing directly down at the component, with the air output about 1-2 inches above the component. Do NOT hold the air gun too close to the PCB; this can burn it. If the air gun blows the component out of place, reduce its air flow and/or hold it higher.

5. Holding the air gun vertically, make small circles with the air gun around the component to heat the PCB area evenly. This is particularly important for heating multiple pins at once.

6. Once the solder paste begins liquifying, do NOT remove the air gun. Continue holding it vertically over the component and making small circles. With your other hand, you may use tweezers to nudge the component into the correct position if needed. Removing the air gun at this point would likely require you to start over.

7. Once the solder paste has completely liquified, the component should settle into the correct position above its pads (to check this, you can very slightly nudge it with the tweezers and see if it returns to the same position afterward). Once the component is oriented in the correct position, remove the air gun. If the component has pads on its underside, you may gently press down on the component while you remove the air gun to ensure proper connection (and hold it in place until the solder solidifies).
8. Visually inspect each of the pins' solder connections with a microscope. Check for shorted pins (due to too much solder) and unconnected pins (due to too little solder). For accessible pins, you can often correct these issues using a soldering iron on the affected pin instead of re-heating the entire component with a hot air gun. If most or all pins are affected, or the affected pins are on the component's underside, you may need to reflow (i.e., reheat to allow the solder to resettle) all pins at once with the air gun. If none of these strategies are successful, consider removing the component (instructions below), cleaning the pads of excess solder, and trying again with a more appropriate amount of solder paste.
9. Once you are happy with the component's solder connections, visually inspect for solder balls and remove them.

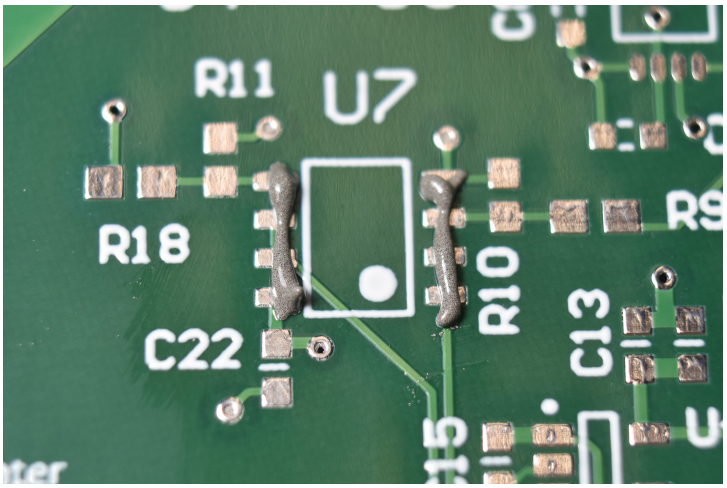


Figure 5: Solder paste applied for an SOIC-8 package IC.

Now, practice removing the IC with the air gun, following the same steps for removing the capacitor above. The strategy of waiting until the component can be "nudged" before attempting to remove it is particularly important for multi-pin components; the solder on each pin will liquefy at different times. If the component does not move when you nudge it, not all pins are liquefied and you should NOT attempt to remove it. Wait until the component moves when you nudge it before attempting to remove it to avoid damaging the PCB pads.

Practice the sequence of soldering the op-amp onto the PCB, inspecting its connections, and removing it until you are comfortable doing each. Be sure to remove excess solder from the pads using solder wick.

Check-off: Demonstrate (a) soldering the component, (b) inspecting the solder connections, and (c) removing the component.

As a note for later, it can be difficult to test all solder connections of an IC or similarly packaged component before you power up the entire circuit. If you experience issues with the component behaving properly when operating the complete circuit, consider reflowing the component.

Cleaning the Board

Once you have finished soldering components onto a PCB, it is best practice to clean any final residue off the board. Since this lab was intended to serve as practice, remove all components from the board when you are finished. Clean the pads of excess solder using solder wick, and clean any remaining residue (often due to flux) off the board using isopropyl alcohol.

Connect the oscilloscope probes to the microcontroller

If you have never used an oscilloscope before or would like to review, tutorials are provided in the form of slides and a short video.

1. **Do not connect the USB cable to the microcontroller before completing the check-off for this section!**
2. Connect an oscilloscope probe to pin 40 on the LaunchPad. This is EPWM1A.
3. Connect an oscilloscope probe to pin 39 on the LaunchPad. This is EPWM1B.
4. Connect the probe reference leads to GND pins on the LaunchPad.

Check-off: Confirm with the instructor that your oscilloscope probes are connected to the LaunchPad correctly. **Do not proceed without completing this check-off. Incorrectly connecting the probe reference leads can cause permanent damage.**

Load the example project onto the microcontroller

1. Connect the USB cable between the LaunchPad and your computer.
2. Right-click the *buck_control_starter* folder in the *Project Explorer* tab, and select *Debug As* → *Code Composer Debug Session*.
3. After the program load has finished, click the *Resume* button in the top of the CCS window.
4. On the oscilloscope, verify that EPWM1A and EPWM1B are complementary with a deadtime of 50 ns.
5. Save a png screenshot of the complementary PWM waveforms clearly showing at least 1 period. You should use the provided USB stick to transfer the file. Do NOT simply take a photo with your cell phone.

Edit the example project to adjust the duty ratio using the CCS debugger

1. Open the file **main.c** within the *buck_control_starter* project.
2. We will declare a 16-bit unsigned integer `duty_cmp` and initialize it to the value `EPWM_CMP_INIT` that we calculated in the pre-lab. This variable holds the counter-compare value passed to the EPWM peripheral. At the top of the file, before `int main(void)`, add the line
   `volatile Uint16 duty_cmp = EPWM_CMP_INIT;` Don't forget the semicolon!
3. Now we will assign the value of `duty_cmp` to the EPWM register that stores the value of the modulating waveform. Find the `while(1)` loop in `main.c`. Add the line `EPwm1Regs.CMPA.bit.CMPA = duty_cmp;`
4. Follow the same steps from before to load the program onto the microcontroller
5. Once the program is running on the microcontroller, navigate to the *Expressions* tab in CCS. Click the *Add new expression* box, and type `duty_cmp`. Click the *Continuous Refresh* button at the top right of the CCS window. This selection enables a periodic refresh of the expressions window, which is useful for seeing a moving snapshot of the values of different variables and expressions.
6. Type in a different value for `duty_cmp` in the *Expressions* tab. Verify that the PWM duty ratio changes when the value of `duty_cmp` is updated in the CCS debugger.
7. Save a png screenshot for a duty ratio other than 50% showing at least 1 period. You should use the provided USB stick to transfer the file. Do NOT simply take a photo with your cell phone.

Add a normalized duty ratio variable

When describing the averaged behavior of power converters, we typically understand duty ratios as normalized variables (in the range 0-1). We will now update our code to include a normalized duty ratio variable `duty` that the rest of the code can interact with.

1. At the top of `main.c`, add a declaration `volatile float duty = 0.5;`
2. In the `while(1)` loop, write some code to now calculate `duty_cmp` from `duty` (Hint: how is the length of a PWM period measured in the microcontroller?)
3. Since `duty` is a normalized variable, we will include saturation in our code. Write some code that limits the minimum and maximum values of the variable `duty` to 0.05 and 0.95 respectively.
4. Load your program onto the microcontroller. Verify that `duty` can be set in the CCS debugger. Verify that the duty ratio never exceeds the limits specified in the previous step.
5. Save a screenshot of your CCS window.

Check-off: (a) Confirm that you can produce complementary PWM waveforms with a switching frequency of 100 kHz, duty ratio of 50%, and deadtime of 50 ns. (b) Demonstrate zooming in and measuring the dead time width on the oscilloscope. (c) Confirm that the normalized duty ratio variable `duty` can be modified in the CCS debugger and results in a corresponding change in the pulse widths. (d) Confirm that the pulse width is clamped when the value of `duty` falls outside the saturation limits. Tell us (e) how much dedicated time you spent on this lab, and (f) if any other student(s) in the lab helped you, and who.

Post-lab Assignment

Lab Results

1. Provide the oscilloscope screenshot showing the complementary PWM waveforms with 50% duty ratio.
2. Provide a snippet of your code showing how `duty_cmp` set in the CCS debugger changes the EPWM CMPA register. Provide the oscilloscope screenshot showing a duty ratio different from 50%.
3. Provide a snippet of your code showing how `duty` set in the CCS debugger changes the EPWM CMPA register. Include a snippet of your code that clearly shows the implementation of the saturation function.

Lab Takeaways

1. In a short paragraph of your own words, describe how to solder through hole components, check their connections, and remove them.
2. In a short paragraph of your own words, describe how to solder surface mount components, inspect their connections, and remove them.
3. Describe the concerns associated with choosing how much solder paste to use on a pad. What happens as a result of too much or too little solder paste?
4. When removing components, why is it important to wait until the component can be "nudged" before taking it off the board?
5. In your own words, describe how analog pulse width modulation is implemented using a comparator. You may include a figure to aid your description.
6. In your own words, contrast the three types of carriers that can be used. What effect do they have on the switching waveform?
7. How does digital PWM differ from analog PWM? What are the relevant registers that must be configured in the C2000 to generate a PWM waveform, and how do they relate to each other?

Build Intuition

1. Describe the tradeoffs associated with choosing a soldering iron temperature.
2. Describe the tradeoffs associated with choosing a hot air gun temperature, air flow rate, and distance from the board. How do these choices affect how long it takes for solder to liquefy?
3. What happens to the digital PWM carrier as the PWM clock frequency is reduced? For a fixed switching frequency, how does this affect the maximum value of the digital PWM counter (i.e TBPRD)? How might this affect a power converter controller?
4. Under what operating conditions can analog and digital PWM be considered nearly identical?

EE 213B

Select a different topology (other than a buck) for your final converter design, and provide justification for why your topology selection is appropriate. This topology should be capable of meeting the same operating point specifications and input and output voltage ripple specifications in Table 1, though you will have an opportunity to choose your switching frequency and sizing requirements for passive components. Your topology must adhere to the following requirements:

- A maximum of eight switches.

- A maximum of two magnetic components.
- A maximum voltage difference of 40 V between *any two nodes* in the circuit.

At the end of this course, we will have competitions for power density (i.e., lowest volume), efficiency, and MPPT. Your converter must be capable of the full specified operating range with thermal stability to be eligible for these. When choosing parts for your converter design, you will be constrained by total board cost. You will be welcome to use any of the parts in our part library (still counting the cost), and there will be an opportunity for you to propose additional part(s) for us to add to this library (though you should count on most parts coming from our existing library).

Assignment Feedback (Required)

How much dedicated time did you spend on this post-lab?