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

EE 113B/213B: Power Electronics Design
Module 5: Sensing and MPPT

Objectives

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

- Size resistors for voltage and current sensing
- Evaluate the features of sensing ICs
- Manage supply rails for gate drive and sensing circuits
- Measure voltages using the ADC module on the C28x Microcontroller
- Implement MPPT using a microcontroller

Pre-lab Assignment

Now that you have your power converter running in open loop, it's time to add closed-loop control (i.e., automatic control of its switching signals). Closed-loop control of a converter's behavior is necessary for it to meet its intended objective as source and load conditions change. One common role of closed-loop control in dc-dc converters is regulating the output voltage to be constant, regardless of the input voltage or load. In your design, you will implement closed-loop control to track the maximum power point of a PV module, allowing the PV module to produce as much power as possible for a given solar irradiance (i.e., solar power). This is known as maximum power point tracking (MPPT).

Implementing MPPT requires knowledge of the PV panel's power output, which can be computed using the power converter's input voltage and current. The C2000 microcontroller has analog-digital converters (ADCs) for sensing such quantities, allowing you to use them as inputs in your control algorithm, but you must first implement sensing circuits to serve as an interfaces between measurement points in your power converter and the ADCs.

Configure the ADC Module

The C28x microcontroller has 4 analog-to-digital converter (ADC) modules called ADCA, ADCB, ADCC, and ADCD. We will refer to these modules generally as "ADCx". The ADC modules can be used to measure voltages and transduced currents in power converters as part of our feedback controllers. Each ADC module can be configured to measure up to 16 different internally multiplexed channels. **We will assume that we want to measure only Channel 2 on each ADC module**. This is determined by the routing on the buck converter PCB.

ADC conversions in the C2000 are configured with a start-of-conversion (SOC) trigger that starts the conversion process. Up to 16 different SOC triggers, numbered SOC0 through SOC15 can be configured. **We will only be using SOC0 in this project**.

The channel to be converted when the SOC0 trigger is received for ADCx is configured via the `AdcxRegs.ADCSOC0CTL.bit.CHSEL` register (i.e., for ADCA, the corresponding register is `AdcaRegs.ADCSOC0CTL.bit.CHSEL`). **We will configure `AdcxRegs.ADCSOC0CTL.bit.CHSEL` on each module to convert Channel 2**.

1. Referring to the Technical Reference Manual on bCourses, determine what value should be set in the register named `ADCSOC0CTL[CHSEL]` to convert channel 2 for every ADC module.
2. In `cpu1_init.c`, update `AdcaRegs.ADCSOC0CTL.bit.CHSEL`, `AdcbRegs.ADCSOC0CTL.bit.CHSEL`, `AdccRegs.ADCSOC0CTL.bit.CHSEL`, and `AdcdRegs.ADCSOC0CTL.bit.CHSEL` correspondingly.

Calculate the measured voltage from the converted result

The ADC module converts a voltage in the range 0 to 3V into a 12-bit unsigned integer value. In this pre-lab, we will call the analog voltage at the input of the ADC `adc_vin` and the converted 12-bit value `conv_result`.

1. Calculate the minimum and maximum values in base-10 that an unsigned 12-bit integer can hold. In this pre-lab, we will call these values `uint12_min` and `uint12_max`.
2. Assuming memory can only be allocated for `conv_result` in multiples of 1 byte, what is the minimum number of bytes needed to store a 12-bit integer? What data type is this size in C?
3. Assuming a linear mapping where 0V at the microcontroller pin corresponds to `uint12_min` and 3V corresponds to `uint12_max`, determine an expression to calculate the `adc_vin` from `conv_result`.

4. Now, assume that when `adc_vin` is 0V, `conv_result` is not `uint12_min` but some other number `conv_offset` (i.e., there is an offset `conv_offset` in the converted ADC result). Determine an expression for `adc_vin` from `conv_result` and `conv_offset`.

5. Now, assume that the voltage at the input of the ADC `adc_vin` is related to the converter output voltage by $\texttt{adc\_vin} = kv_{\text{out}}$. Determine an expression for v_{out} from `conv_result`.

Voltage Sensing

Voltage sensing is commonly implemented using resistive voltage dividers that step the voltage down to a range compatible with the ADC. Then, an op-amp configured to serve as a unity-gain buffer is commonly added between the divider and the ADC to prevent the ADC's input impedance from affecting the voltage division. This configuration is shown in Fig. 1.

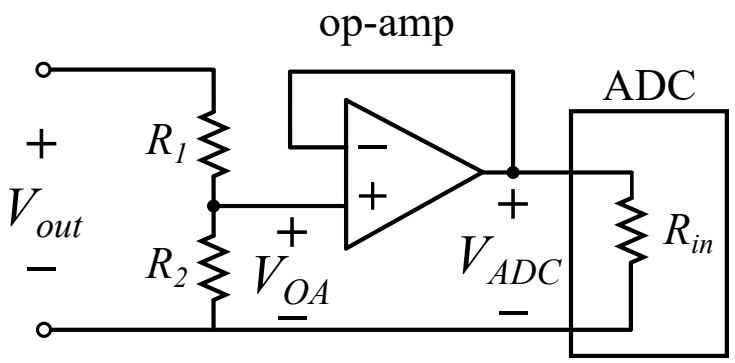


Figure 1: Voltage sensing circuit.

In your power converter, assume you would like to sense the output voltage.

1. Design a resistive voltage divider (i.e., choose resistor values) that steps the full voltage range of interest down to a voltage range that can be accommodated by the C2000 ADCs. You are encouraged to choose from standard resistor values in our parts library.

2. Calculate the voltage divider's power consumption. If needed, select new resistor values such that the divider's power consumption is orders of magnitude below the most significant expected losses in your power converter.

3. Assume you will implement the buffer with the 2-channel op-amp provided in the parts library. Confirm that the op-amp's input bias current is negligible compared to the voltage divider's expected current. This is important for ensuring the presence of the op-amp will not interfere with the voltage division.

Current Sensing

One strategy for current sensing involves adding small sense resistors directly in the current paths of interest and measuring their resulting voltages. This is commonly done using current sense amplifiers, which provide high-precision differential voltage measurements. For your design, assume you will use the current sense amplifier IC provided in our parts library. Explore the important features of this IC as follows:

1. Identify the gain of the current sense amplifier.

2. While this current sense amplifier's output is referenced to ground, its input is a differential measurement (i.e., not referenced to ground). Identify the absolute maximum voltage differences between (a) the two input pins, and (b) one or more input pin and ground. Comment on whether or not these ratings are suitable for the voltages these input pins will experience in your power converter design.

Now, assume you would like to sense the output current.

1. Select a sense resistance value from our parts library that is compatible with the input voltage ranges of both the current sense amplifier and the C2000 ADCs (considering the amplifier's gain). To ensure high signal integrity, it is best practice to choose a sense resistance value that will utilize most of these available voltage ranges. You should consider all current levels to be sensed throughout the converter's full operating range.

2. Calculate the sense resistor's power consumption and evaluate it with respect to the most significant losses in your converter. If needed, select a new resistance value such that the sense resistor's power consumption is negligible compared to other losses. If you do select a new resistance value, evaluate how much of the sensing circuit input voltage ranges will be utilized, keeping in mind that wide use of this range is needed for signal integrity. You will observe there is a tradeoff between power consumption and signal integrity.

Full Sensing Circuit

Now that you have designed your voltage divider(s) and sense resistors, finalize your sensing circuit as follows:

1. Identify the supply voltage requirements for the op-amp and current sense amplifier, and select a single supply voltage that is appropriate for both of them. If this voltage is different than the supply voltage level for your gate drive circuit, choose a linear regulator (or LDO) from the parts library capable of converting the gate circuit supply voltage to your selected sensing circuit supply voltage.

2. Given this supply voltage, calculate the output reference voltage (i.e., its "output common mode level") generated at the current sense amplifier's REF pin, according to its datasheet. This will be the baseline voltage that the amplifier outputs when it measures zero current flowing through the sense resistor. For positive and negative current readings, the amplifier's output voltage will rise above and below the reference, respectively. Thus, compare this reference voltage with the maximum C2000 ADC voltage input. If the reference is greater than half the ADC voltage limit, it is wise to manually decrease it by connecting an external voltage reference IC of lower voltage; this will give your amplifier a greater portion of the ADC input voltage range for providing precise current measurements that are greater than zero. If needed, choose a voltage reference IC from our part library that has a voltage value of less than half the ADC's maximum voltage. You will later connect this voltage reference IC directly to the current sense amplifier's REF pin to override its internally generated reference.

3. For each IC in your sensing circuit (op-amp, current amplifier, LDO, voltage reference, etc.), describe where decoupling capacitor(s) are needed, if any. Draw a circuit diagram of how you plan to connect these decoupling capacitor(s) to the pins of each. Size your decoupling capacitor with parts in our library assuming an 0603 MLCC package, using the values suggested in the IC datasheets or higher.

Assignment Feedback (Required)

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

Lab Safety

By continuing this module, you agree to the following important safety rules in the lab:

1. Always wear safety glasses when you, or any surrounding workbenches, have your main power supply on (i.e., the power supply acting as your converter's voltage/current source). Power converter failures often involve small projectiles that can cause serious damage to your eyes.
2. Do not wear metal jewelry on your hands or wrists, including watches, while testing your power converter.
3. Do not increase any power supply voltage above 30 V for any reason. To ensure a power supply does not surpass this limit when operating in constant-current mode, ensure its internal voltage limit is set to 30 V (or lower, to prevent your converter from failing).
4. Do not touch your converter configuration, even with probes, when your main power supply is on. Get in the habit of arranging your probes how you would like them before turning on the main power supply. "Live probing" is reserved for TAs only.
5. Make sure all wires and loose pieces are far from you converter, and that the converter itself and all connected cables are far from your body, before you turn on the main power supply. Position your converter in a location where you will not have to reach directly over it to adjust equipment.
6. Adopt a "risk minimization" mindset when configuring your test setup and test procedure. For example, make sure no wires are hanging off the bench in a way that you could bump them. If you are concerned about bumping certain cables, tape them down.
7. If at any point your converter begins to malfunction, turn off the main power supply if it is safe to do so. This should be your first reaction to stop testing.
8. If at any point you must work with an energized circuit (e.g., the gate circuit), use only one hand and make sure the rest of your body isn't touching any conductive surfaces.
9. Keep your converter on the rubber heat mat at all times while soldering and testing.
10. Keep your workbench is clean and free of other objects as possible. Use the lab computer instead of your own if you need to free up space.
11. Never touch the heated surface of a soldering iron or hot air gun.
12. Always use the solder fan when soldering to protect yourself from fumes.
13. Be sure to appropriately shield heat-sensitive components (e.g., plastic components, ICs, and electrolytic capacitors) with foil when using the hot air gun for soldering. Some components (e.g., electrolytic capacitors) may explode if exposed to heat.
14. Some types of solder contain Lead. Always wash your hands after soldering, especially before eating or drinking, and beware of contaminating personal property.

If you have any questions, please reach out to course staff before proceeding.

Lab Assignment

Read Converted ADC Results

Configure the 6V channel of the low-voltage power supply to output a voltage of **0.5 V** with a current limit of 0.1 A. **Ensure your power supply is set to the correct voltage before proceeding. Otherwise, you risk permanently damaging the microcontroller's ADC.** The ADC conversion result is stored as a 16-bit value in an ADC result register.

1. Verify that the output of the bench power supply is **disabled** before proceeding.
2. For each ADC, a conversion result register `AdcxResultRegs.ADCRESULT0` is updated when the conversion corresponding to SOC0 is complete. In `main.c` write code to compute the voltage at the input of ADCA Channel 2 (in the pre-lab, we called this variable `adc_vin`) from the value of the conversion result register `AdcaResultRegs.ADCRESULT0`. You may assume there is no offset.
3. Load your program onto the microcontroller.
4. Take one red and one black banana-to-hook cable from the wire rack. Connect the red hook to pin 29 on the LaunchPad. Connect the black hook to a GND pin on the LaunchPad.
5. After verifying your connections, connect the banana plugs to the 6V-channel of the power supply. After verifying all connections, enable the output of the power supply **(which should still be set to 0.5 V)**.
6. In the CCS debugger, view the calculated voltage by adding the appropriate expression to the *Expressions* tab.
7. Save a screenshot (using a lab provided usb) of your CCS window showing the variable corresponding to the calculated voltage.
8. Reconfigure the power supply to output 1 V. Save a screenshot of your CCS window showing the variable corresponding to the calculated voltage.

Calibrate ADC Gain and Offset

The accuracy of your measurement can improve by calibrating the ADC. Calibration involves comparing measured results to known values and creating an expression to relate the two. Each ADC module may have different calibration values.

1. Sweep the power supply output voltage from 0 V to 3 V with a step size of 0.25 V. Record the corresponding value of `AdcaResultRegs.ADCRESULT0` for each step.
2. Find a line of best fit for the recorded data points.
3. Write code to compute the voltage at the input of ADCA Channel 2 using the gain and offset determined by the line of best fit.
4. Load the program onto the microcontroller. Reconfigure your power supply voltage to 0.5 V, and validate your new calculated value. Likewise confirm the new calculated value for a power supply voltage of 1 V.

Check-off: (a) Discuss how well the calculated value in the CCS debugger matched the applied voltages of 0.5 V and 1 V before calibration (in the previous section). (b) Discuss how well the calculated value in the CCS debugger matched the applied voltages of 0.5 V and 1 V after calibration. Do they match better than the calculated value found prior to calibration?

✓ It's OK

Iout: pin27 Vout: pin29

Implement Sensing Circuit

The past few modules have guided you through several examples of systematically assembling and testing different sections of your power converter prototype, ensuring proper operation of each new addition before proceeding. Now, apply these principles to systematically assemble and test sensing circuits for output voltage and output current. Once fully operational, your microcontroller should correctly sense these values and correctly calculate the output power of the converter. Refer to the circuit schematic, parts library, LaunchPad user guide, and component datasheets for guidance.

Check-off: Confirm microcontroller is correctly sensing output voltage and output current, with correct power calculations, for two different operating points within the converter's specified operating region.

Implement MPPT in Microcontroller Code

Download the provided MPPT_Skeleton.c file from bCourses. This file contains example code written for sweeping over the entire duty cycle range to find the PV module's maximum power point. This file does not contain many of the configuration settings and initialization necessary for the project to build; refer to your previous code for this information. Modify MPPT_Skeleton.c for MPPT as follows:

1. Where prompted, implement ADC voltage/current conversions and add code for measuring and calculating output power.
2. Where prompted, implement a basic "Perturb and Observe" MPPT algorithm. After reading voltage/current and calculating power, your code will have to make a decision on how to adjust the converter's duty cycle to get closer to the maximum power point. You are encouraged to follow the instructions in the code's comments.
3. Where prompted, add code for updating the EPWM register.

Once you have implemented your MPPT, use the CCS debugger to emulate sensed measurements and confirm that your MPPT algorithm moves the duty cycle in the appropriate direction. Ensure you have a simple way to enable/disable MPPT vs. a programmed duty cycle.

Enable MPPT

At this point, you have independently verified the operation of your power converter, your sensing circuitry, and your MPPT algorithm. Now, it's time to tie them all together!

1. Set up PV module emulation at the converter's input as conducted in the previous module. Keep the electronic load in constant voltage mode at 12 V, and configure the dc power supply to emulate full I_{SC}. Operate the power converter at a constant duty cycle close to (but not exactly at) the PV panel's known maximum power point.
2. Enable the microcontroller's MPPT, and observe its behavior as measured by your sensing circuit (displayed in the CCS debugger). If it does not correctly track the maximum power point, disable the MPPT and troubleshoot individual sections of the loop (sensing, microcontroller code, PWM, gate drive, power converter, etc.).

Once your MPPT functions correctly, compare the MPPT it locates at full I_{SC} (as measured by your sensing circuit) to the value you manually identified in the previous module. Then, toggle the emulated I_{SC} between its full and half values, and other values in between. Your MPPT algorithm should eventually find the maximum achievable power point in all cases.

Check-off: (a) Demonstrate MPPT functioning appropriately at full I_{SC} as measured by your sensing circuit, (b) compare the observed maximum power point with the point you manually identified in the previous module, and (c) demonstrate MPPT locating the new achievable maximum when you toggle I_{SC} between its half and full

values. Tell us (d) how much dedicated time you spent on this lab, and (e) if any other student(s) in the lab helped you, and who.

Post-lab Assignment

Lab Results

1. Provide a snippet of your code showing the calculation of the voltage at the input of the ADC pin. Provide a screenshot of the CCS debugger showing the calculated voltage.
2. Provide a screenshot of your MPPT algorithm implemented in code.
3. Provide the maximum power point identified by your control for both full and half I_{SC}. For each point, provide the duty cycle, input voltage, and output power as measured by your microcontroller.

Lab Takeaways

1. In your own words, describe how voltage sensing can be designed and implemented to provide feedback to a controller.
2. In your own words, describe how current sensing can be designed and implemented to provide feedback to a controller.
3. Describe how the various supply voltage needs of gate drive and sensing circuits can be met with minimal external supplies.
4. In your own words, describe how to calibrate an ADC and why it is necessary.
5. In your own words, describe how closed-loop control may be implemented using a microcontroller.

Build Intuition

1. Compare voltage sensing and current sensing in terms of cost, complexity, and loss. If you could only implement one in a converter design, which would you choose and why?
2. Describe how certain types of sources and loads may be emulated in a laboratory setting with power supplies and electronic loads. Provide two examples of sources that you would emulate in different ways, and how you would emulate them. Provide two examples of loads that you would emulate in different ways, and how you would emulate them.
3. Describe how your MPPT algorithm would behave if your electronic load was configured to emulated a constant-resistance load. Would it still work? If so, what would be different?

EE 213B

Design the gate drive circuit(s) for your converter design. This should include selecting gate driver(s), selecting digital isolators (if applicable), designing bootstrap circuits (if applicable), and placing decoupling capacitors. You should select all necessary components (besides gate resistor values - these can be tuned when you power up) from our parts library, or propose new parts to be added according to the procedure outlined in Module _.

Then, design the sensing circuit(s) needed for MPPT in your converter design. This should include sense resistor sizing and selection of all necessary ICs from our parts library.

Finalize your power stage design review slides, and add your gate drive and sensing circuits to the end of your slide deck as follows:

Gate Drive Circuit:

1. The gate drive configurations you plan to use for each switch (half-bridge driver, low-side driver, low-side driver with isolator, etc.).

2. Schematic of your gate drive circuit(s).
3. Part numbers and relevant ratings for each component of your gate drive circuit.

Sensing Circuit:

1. What you plan to sense.
2. Schematic of your sensing circuit(s).
3. Part numbers and relevant ratings for each component of your sensing circuit(s).

Bring your slides to your design review appointment.

Assignment Feedback (Required)

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

1. Referring to the Technical Reference Manual on bCourses, determine what value should be set in the register named `ADCSOCOCTL[CHSEL]` to convert channel 2 for every ADC module.

Select input channel.

2. In `cpu1_init.c`, update `AdcaRegs.ADCSOCOCTL.bit.CHSEL`, `AdcbRegs.ADCSOCOCTL.bit.CHSEL`, `AdccRegs.ADCSOCOCTL.bit.CHSEL`, and `AdcdRegs.ADCSOCOCTL.bit.CHSEL` correspondingly. ⇒ All to Channel 2.

1.
```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 2;
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 2;
AdccRegs.ADCSOC0CTL.bit.CHSEL = 2;
AdcdRegs.ADCSOC0CTL.bit.CHSEL = 2;
```

2.

Table 11-6. Channel Selection of Input Pins

Input Mode	CHSEL	Input	
Single-Ended	0	ADCIN0	
	1	ADCIN1	
	2	ADCIN2	
	3	ADCIN3	
	4	ADCIN4	
	5	ADCIN5	
	6	ADCIN6	
	7	ADCIN7	
	8	ADCIN8	
	9	ADCIN9	
	10	ADCIN10	
	11	ADCIN11	
	12	ADCIN12	
	13	ADCIN13	
	14	ADCIN14	
	15	ADCIN15	
	CHSEL	**Positive Input**	**Negative Input**
Differential	0 or 1	ADCIN0	ADCIN1
	2 or 3	ADCIN2	ADCIN3
	4 or 5	ADCIN4	ADCIN5
	6 or 7	ADCIN6	ADCIN7
	8 or 9	ADCIN8	ADCIN9
	10 or 11	ADCIN10	ADCIN11
	12 or 13	ADCIN12	ADCIN13
	14 or 15	ADCIN14	ADCIN15

```
// ADCA Configuration

AdcaRegs.ADCCTL2.bit.PRESCALE = 6;
AdcSetMode(ADC_ADCA, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 1;
AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;
DELAY_US(1000);

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 2;
AdcaRegs.ADCSOC0CTL.bit.ACQPS = ADC_ACQ_WINDOW;
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 5;

AdcaRegs.ADCSOCPRICTL.bit.SOCPRIORITY = 1;

AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 0;
AdcaRegs.ADCINTSEL1N2.bit.INT1CONT = 1;
AdcaRegs.ADCINTSEL1N2.bit.INT1E = 0;
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
```

← For ADC A,B,C,D

Calculate the measured voltage from the converted result

The ADC module converts a voltage in the range 0 to 3V into a 12-bit unsigned integer value. In this pre-lab, we will call the analog voltage at the input of the ADC `adc_vin` and the converted 12-bit value `conv_result`.

1. Calculate the minimum and maximum values in base-10 that an unsigned 12-bit integer can hold. In this pre-lab, we will call these values `uint12_min` and `uint12_max`.

⇕ 8 bits

2. Assuming memory can only be allocated for `conv_result` in multiples of 1 byte, what is the minimum number of bytes needed to store a 12-bit integer? What data type is this size in C?

3. Assuming a linear mapping where 0V at the microcontroller pin corresponds to `uint12_min` and 3V corresponds to `uint12_max`, determine an expression to calculate the `adc_vin` from `conv_result`.

(1)
uint12_min = 0
uint12_max = $2^{12}-1$

(2) 1 byte ⟺ 8 bits
12 bits ← needs 2 bytes
2 byte ⇒ uint16_t.

(3).
```
adc_vin = conv_result * 3 / 4095
```
uint16_t

4. Now, assume that when `adc_vin` is 0V, `conv_result` is not `uint12_min` but some other number `conv_offset` (i.e., there is an offset `conv_offset` in the converted ADC result). Determine an expression for `adc_vin` from `conv_result` and `conv_offset`.

5. Now, assume that the voltage at the input of the ADC `adc_vin` is related to the converter output voltage by $\texttt{adc\_vin} = k v_{\text{out}}$. Determine an expression for v_{out} from `conv_result`.

(4).

adc_vin:	conv_result
0V	conv_offset
3V	4095+ conv_offset

$\Rightarrow$ conv_result − conv_offset = adc_vin * 4095 / 3

$\Rightarrow$ adc_vin = (conv_result − conv_offset) * 3 / 4095.

(5). $V_{out} = \frac{1}{k}\, adc\_vin = \frac{3}{k \cdot 4095} \cdot (conv\_result - conv\_offset)$.

Voltage Sensing

Voltage sensing is commonly implemented using resistive voltage dividers that step the voltage down to a range compatible with the ADC. Then, an op-amp configured to serve as a unity-gain buffer is commonly added between the divider and the ADC to prevent the ADC's input impedance from affecting the voltage division. This configuration is shown in Fig. 1.

? what do I measure.

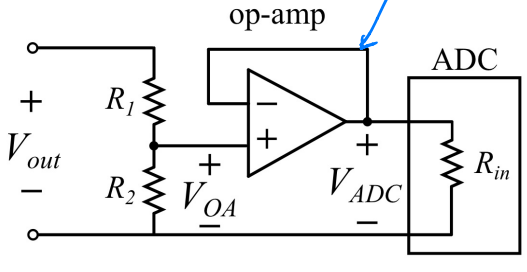


Figure 1: Voltage sensing circuit.

In your power converter, assume you would like to sense the output voltage.

1. Design a resistive voltage divider (i.e., choose resistor values) that steps the full voltage range of interest down to a voltage range that can be accommodated by the C2000 ADCs. You are encouraged to choose from standard resistor values in our parts library.

2. Calculate the voltage divider's power consumption. If needed, select new resistor values such that the divider's power consumption is orders of magnitude below the most significant expected losses in your power converter.

3. Assume you will implement the buffer with the 2-channel op-amp provided in the parts library. Confirm that the op-amp's input bias current is negligible compared to the voltage divider's expected current. This is important for ensuring the presence of the op-amp will not interfere with the voltage division.

1. $V_{out} = 12V$ (fixed) input ADC from C2000: 0~3V

ripple ≤ 5%

range: [11.4V, 12.6V]

select $R_2 = 100k\Omega$

$R_1 = 470k\Omega$

resistor dividers, 100k, 0603	ERJ-PB3B1003V
resistor dividers, 470k, 0603	RN73R1JTTD4703B25

2. Calculate the voltage divider's power consumption. If needed, select new resistor values such that the divider's power consumption is orders of magnitude below the most significant expected losses in your power converter.

power consumption: [neglecting OPAMP driving loss]

$$P \approx \frac{V_{out}^2}{R_1+R_2} = \frac{12^2}{(470+100)k\Omega} \approx 2.53\times10^{-4} W \ll P_{loss} = 5W \text{ at worst case.}$$

3. Assume you will implement the buffer with the 2-channel op-amp provided in the parts library. Confirm that the op-amp's input bias current is negligible compared to the voltage divider's expected current. This is important for ensuring the presence of the op-amp will not interfere with the voltage division.

LM358DR:

Family Comparison

Specification	LM358B LM358BA	LM2904B LM2904BA	LM358 LM358A	LM2904	LM2904V LM2904AV	LM258 LM258A	LM158 LM158A	Units
Supply voltage	3 to 36	3 to 36	3 to 30	3 to 26	3 to 30	3 to 30	3 to 30	V
Offset voltage (max, 25°C)	± 3 ± 2	± 3 ± 2	± 7 ± 3	± 7	± 7 ± 2	± 5 ± 3	± 5 ± 2	mV
Input bias current (typ / max)	10 / 35	10 / 35	20 / 250 15 / 100	20 / 250	20 / 250	20 / 150 15 / 80	20 / 150 15 / 50	nA
Gain bandwidth product	1.2	1.2	0.7	0.7	0.7	0.7	0.7	MHz
Supply current (typ, per channel)	0.3	0.3	0.35	0.35	0.35	0.35	0.35	mA
ESD (HBM)	2000	2000	500	500	500	500	500	V
Operating ambient temperature	−40 to 85	−40 to 125	0 to 70	−40 to 125	−40 to 125	−25 to 85	−55 to 125	°C

using worst-case: 35nA & 250nA.

compared to $I_{R_1-R_2} = \frac{V_{out}}{R_1+R_2} = \frac{12V}{(470k+100k)\Omega} \approx 2.1\times10^{-5} = 21\mu A \gg (250nA) \text{ or } 35nA$

s.t. input current bias is neglible.

Current Sensing

One strategy for current sensing involves adding small sense resistors directly in the current paths of interest and measuring their resulting voltages. This is commonly done using current sense amplifiers, which provide high-precision differential voltage measurements. For your design, assume you will use the current sense amplifier IC provided in our parts library. Explore the important features of this IC as follows:

1. Identify the gain of the current sense amplifier.
2. While this current sense amplifier's output is referenced to ground, its input is a differential measurement (i.e., not referenced to ground). Identify the absolute maximum voltage differences between (a) the two input pins, and (b) one or more input pin and ground. Comment on whether or not these ratings are suitable for the voltages these input pins will experience in your power converter design.

Now, assume you would like to sense the output current.

1. Library: LT999-50

A_V	Gain	LT1999-10	●	9.95	10	10.05	V/V
		LT1999-20	●	19.9	20	20.1	V/V
		LT1999-50	●	49.75	50	50.25	V/V

2. From the library, max $R_{sense} = 5m\Omega$,

(a) $I_{out} \cdot R_{sense} = V_{diff\,max} = \frac{25}{3} \times 5m\Omega = 0.0417V$

(b) $V_{input \to gnd\,max} = 12V + 0.0417V = 12.0417V$

ABSOLUTE MAXIMUM RATINGS (Note 1)

Differential Input Voltage
+IN to –IN (Notes 1, 3) ..... ±60V, 10ms
+IN to GND, –IN to GND (Note 2) ..... –5.25V to 88V
Total Supply Voltage (V^+ to GND) 6V
Input Voltage Pins 6 and 8 ..... V^+ + 0.3V, –0.3V
Output Short-Circuit Duration (Note 4) ..... Indefinite
Operating Ambient Temperature (Note 5)
LT1999C ..... –40°C to 85°C
LT1999I ..... –40°C to 85°C
LT1999H ..... –40°C to 125°C
LT1999MP ..... –55°C to 150°C

Specified Temperature Range (Note 6)
LT1999C ..... 0°C to 70°C
LT1999I ..... –40°C to 85°C
LT1999H ..... –40°C to 125°C
LT1999MP ..... –55°C to 150°C
Junction Temperature ..... 150°C
Storage Temperature Range ..... –65°C to 150°C

ratings: ±60V differential input
IN→ GND: −5.25~88V ✓ suitable

1. Select a sense resistance value from our parts library that is compatible with the input voltage ranges of both the current sense amplifier and the C2000 ADCs (considering the amplifier's gain). To ensure high signal integrity, it is best practice to choose a sense resistance value that will utilize most of these available voltage ranges. You should consider all current levels to be sensed throughout the converter's full operating range.

2. Calculate the sense resistor's power consumption and evaluate it with respect to the most significant losses in your converter. If needed, select a new resistance value such that the sense resistor's power consumption is negligible compared to other losses. If you do select a new resistance value, evaluate how much of the sensing circuit input voltage ranges will be utilized, keeping in mind that wide use of this range is needed for signal integrity. You will observe there is a tradeoff between power consumption and signal integrity.

(1)
For maximum power, suppose $P_{out} \in [50W, 100W]$, $I_{out} \in [\frac{25}{6}A, \frac{25}{3}A]$. $A_V = 50$

~~5mΩ~~ 2.5mΩ → $V_{sense} \in$ [~~20.83mV, 41.66mV~~ 10.42mV, 20.84mV]

current ⇒ OP AMP: V_{out}: [~~1.04V, 2.08V~~ 0.52V, 1.04V] + V_{REF} = 1.25V ⇒ [1.77, 2.29V]

(2). $P_{loss} = I_{out}^2 \cdot R_{sense} \simeq 0.347W$. $P_{loss\_conduction} \simeq 1.8W$.

⇒ It's about 20% of the most significant loss.

not negligible, but compared to power loss, signal integrity is much more important.

Full Sensing Circuit

Now that you have designed your voltage divider(s) and sense resistors, finalize your sensing circuit as follows:

1. Identify the supply voltage requirements for the op-amp and current sense amplifier, and select a single supply voltage that is appropriate for both of them. If this voltage is different than the supply voltage level for your gate drive circuit, choose a linear regulator (or LDO) from the parts library capable of converting the gate circuit supply voltage to your selected sensing circuit supply voltage.
2. Given this supply voltage, calculate the output reference voltage (i.e., its "output common mode level") generated at the current sense amplifier's REF pin according to its datasheet. This will be the baseline voltage that the amplifier outputs when it measures zero current flowing through the sense resistor. For positive and negative current readings, the amplifier's output voltage will rise above and below the reference, respectively. Thus, compare this reference voltage with the maximum C2000 ADC voltage input. If the reference is greater than half the ADC voltage limit, it is wise to manually decrease it by connecting an external voltage reference IC of lower voltage; this will give your amplifier a greater portion of the ADC input voltage range for providing precise current measurements that are greater than zero. If needed, choose a voltage reference IC from our part library that has a voltage value of less than half the ADC's maximum voltage. You will later connect this voltage reference IC directly to the current sense amplifier's REF pin to override its internally generated reference.
3. For each IC in your sensing circuit (op-amp, current amplifier, LDO, voltage reference, etc.), describe where decoupling capacitor(s) are needed, if any. Draw a circuit diagram of how you plan to connect these decoupling capacitor(s) to the pins of each. Size your decoupling capacitor with parts in our library

Family Comparison

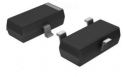
Specification	LM358B LM358BA	LM2904B LM2904BA	LM358 LM358A	LM2904	LM2904V LM2904AV	LM258 LM258A	LM158 LM158A	Units
Supply voltage	3 to 36	3 to 36	3 to 30	3 to 26	3 to 30	3 to 30	3 to 30	V
Offset voltage (max, 25°C)	± 3 ± 2	± 3 ± 2	± 7 ± 3	± 7	± 7 ± 2	± 5 ± 3	± 5 ± 2	mV
Input bias current (typ / max)	10 / 35	10 / 35	20 / 250 15 / 100	20 / 250	20 / 250	20 / 150 15 / 80	20 / 150 15 / 50	nA
Gain bandwidth product	1.2	1.2	0.7	0.7	0.7	0.7	0.7	MHz
Supply current (typ, per channel)	0.3	0.3	0.35	0.35	0.35	0.35	0.35	mA
ESD (HBM)	2000	2000	500	500	500	500	500	V
Operating ambient temperature	−40 to 85	−40 to 125	0 to 70	−40 to 125	−40 to 125	−25 to 85	−55 to 125	°C

V_S	Supply Voltage (Note 11)	●	4.5	5	5.5	V

V_{REF}	Open Circuit Voltage	$V_{\overline{SHDN}}$ = 0.5V	● 2.45	2.5	2.55	V
			● 1	2.5	2.75	V

Voltage - Input (Max)	16V
Voltage - Output (Min/Fixed)	5V
Voltage - Output (Max)	-
Voltage Dropout (Max)	0.58V @ 150mA
Current - Output	150mA
Current - Quiescent (Iq)	95 µA
Current - Supply (Max)	2.5 mA

ADR1581ARTZ-REEL7	
DigiKey Part Number	505-ADR1581ARTZ-REEL7TR-ND - Tape & Reel (TR) 505-ADR1581ARTZ-REEL7CT-ND - Cut Tape (CT) 505-ADR1581ARTZ-REEL7DKR-ND - Digi-Reel®
Manufacturer	Analog Devices Inc.
Manufacturer Product Number	ADR1581ARTZ-REEL7
Description	IC VREF SHUNT 0.12% SOT23-3
Manufacturer Standard Lead Time	10 Weeks



3. For each IC in your sensing circuit (op-amp, current amplifier, LDO, voltage reference, etc.), describe where decoupling capacitor(s) are needed, if any. Draw a circuit diagram of how you plan to connect these decoupling capacitor(s) to the pins of each. Size your decoupling capacitor with parts in our library assuming an 0603 MLCC package, using the values suggested in the IC datasheets or higher.

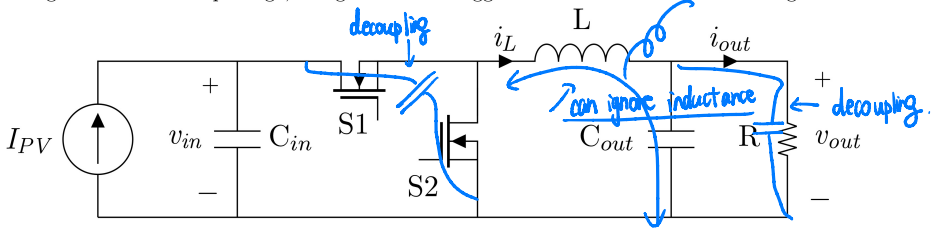


Figure 1: Synchronous buck converter.

Actually, the decoupling capacitor [because of the commutation loop] can only be placed at Cin & Cout.

with the inductance L, we don't have to use decoupling capacitor larger enough, (ignoring L-par)

We can choose decoupling cap. for gate driver and 2.2μF For Vout.
↑
10μF

Feedback: 4.5 hrs.

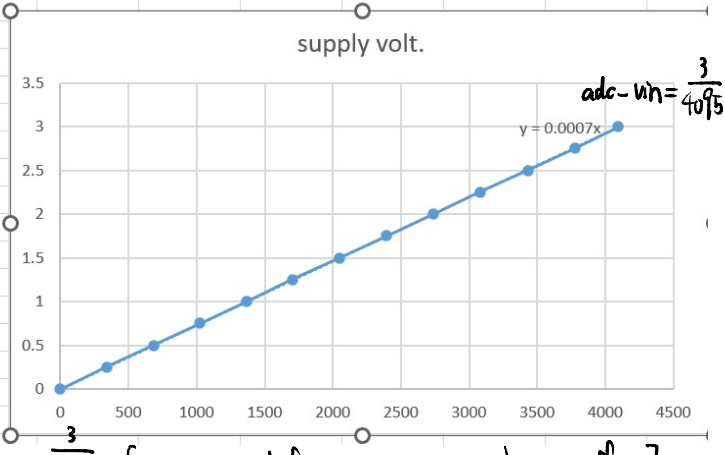
0.5V input ADC.

duty_cmp	unsigned int	240	0x0000AA39
duty	float	0.5	0x0000AA3E
deadtime_rise	unsigned int	5	0x0000AA3A
deadtime_fall	unsigned int	5	0x0000AA3B
adc_vin	float	0.504761875	0x0000AA3C
adc_raw	unsigned int	693	0x0000AA38
Add new expre.			

1V input ADC.

duty_cmp	unsigned int	240	0x0000AA39
duty	float	0.5	0x0000AA3E
deadtime_rise	unsigned int	5	0x0000AA3A
deadtime_fall	unsigned int	5	0x0000AA3B
adc_vin	float	1.0029304	0x0000AA3C
adc_raw	unsigned int	1369	0x0000AA38
Add new expre.			

ADCRESULT0	supply volt.
0	0
340	0.25
685	0.5
1025	0.75
1365	1
1701	1.25
2050	1.5
2395	1.75
2735	2
3080	2.25
3430	2.5
3775	2.75
4095	3



$adc\_vin = \frac{3}{4095} \cdot RESULT$

formula: $adc\_vin = adc\_raw \cdot \frac{3}{4095}$. [same as before, accurate enough, no offset]

post-lab:

Lab Results

1. Provide a snippet of your code showing the calculation of the voltage at the input of the ADC pin. Provide a screenshot of the CCS debugger showing the calculated voltage.

2. Provide a screenshot of your MPPT algorithm implemented in code.

3. Provide the maximum power point identified by your control for both full and half I_{SC}. For each point, provide the duty cycle, input voltage, and output power as measured by your microcontroller.

```
int main(void)
{
    InitSysCtrl(); // Initialize SYSCTL (PLL, Watchdog, etc.)

#ifdef LAUNCHPAD // include this define in project settings if using a LaunchPad
    EALLOW;
    ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV = 0; // remove /2 clock division for LaunchPad to make calculations consistent with ControlCard
    EDIS;
#endif

    InitGpio(); // Initialize GPIO register states

    configure_GPIO(); // configure GPIO settings

    DINT;       // Disable ST1.INTM
    IER = 0x0000; // Disable CPU interrupts
    IFR = 0x0000; // Clear all CPU interrupt flags

    InitPieCtrl();    // Initialize PIE control registers
    InitPieVectTable(); // Initialize PIE vector table to default ISR locations...
                        // This will typically be overwritten later

    configure_ADC();  // configure ADC settings
    configure_EPWM(); // configure EPWM settings

    while (1) // main circuit setup
    {
        // 2 Mode ctrl:
        // For safety concern...
        if (duty &lt;= 0.55)
        {
            duty = 0.55;
        }
        else if (duty &gt;= 0.75)
        {
            duty = 0.75;
        }
        // Refresh duty&amp;duty_cmp
        duty_cmp = EPWM_TBPRD * duty;
        EPwm1Regs.CMPA.bit.CMPA = duty_cmp;

        EPwm1Regs.DBRED.bit.DBRED = deadtime_rise;
        EPwm1Regs.DBFED.bit.DBFED = deadtime_fall;

        // ADC read and conversion
        adc_raw_vout = AdcaResultRegs.ADCRESULT0;
        adc_vout = ((float)adc_raw_vout / 4095.0) * 3.0 * 11; // float from 12-bit value to 3V

        adc_raw_iout = AdccResultRegs.ADCRESULT0;
        adc_iout = ((float)adc_raw_iout * 0.006189 - 10.7);

        DELAY_US(wait_time); // Insert a delay longer than the switching period to ensure that the duty ratio
                             //   is updated at most once per switching period (we are implementing single-update PWM)

        if (!flag_sweep){
            stop_search = 0;
            Perturb_Observe = 0;
        }

        if (!stop_search &amp;&amp; flag_sweep)
        {
            //Sweep over the entire duty cycle range to find the maximum power point
            //Wait some time to allow the system to settle
            DELAY_US(wait_time);
            //Done: Measure Output Power
            pout = adc_iout * adc_vout;
            MPP_Duty = duty;
            MPP_power = pout;
            stop_search = 1;
            Perturb_Observe = 1;
        }

        // Perturb Observe: give a perturb to see changes
        if(Perturb_Observe &amp;&amp; flag_sweep){
            //Done: Implement Perturb and Observe Algorithm
            // Refresh duty&amp;duty_cmp ***current is MPPT***
            duty = MPP_Duty;
            duty_cmp = EPWM_TBPRD * duty;
            EPwm1Regs.CMPA.bit.CMPA = duty_cmp;
            DELAY_US(wait_time);

            adc_raw_vout = AdcaResultRegs.ADCRESULT0;
            adc_vout = ((float)adc_raw_vout / 4095.0) * 3.0 * 11; // float from 12-bit value to 3V

            adc_raw_iout = AdccResultRegs.ADCRESULT0;
            adc_iout = ((float)adc_raw_iout * 0.006189 - 10.7);

            MPP_power = adc_iout * adc_vout; // Set current status as MPPT ****

            duty_inc = MPP_Duty + duty_step;
            duty_dec = MPP_Duty - duty_step;
            // Test duty up and down -- Always down, up, down, up...
            if (flag_D_dec == 1 &amp;&amp;&amp; flag_D_inc == 1){
                flag_D_dec = 0;
                flag_D_inc = 0;
            }
            else if (flag_D_dec == 0 &amp;&amp;&amp; flag_D_inc == 0){
                duty = duty_dec;
                flag_D_dec = 1;
                flag_D_inc = 0;
            }
            else if (flag_D_dec == 1 &amp;&amp;&amp; flag_D_inc == 0){
                duty = duty_inc;
                flag_D_inc = 1;
                flag_D_dec = 1;
            }
            // Refresh duty&amp;duty_cmp
            duty_cmp = EPWM_TBPRD * duty;
            EPwm1Regs.CMPA.bit.CMPA = duty_cmp;

            DELAY_US(wait_time);

            adc_raw_vout = AdcaResultRegs.ADCRESULT0;
            adc_vout = ((float)adc_raw_vout / 4095.0) * 3.0 * 11; // float from 12-bit value to 3V

            adc_raw_iout = AdccResultRegs.ADCRESULT0;
            adc_iout = ((float)adc_raw_iout * 0.006189 - 10.7);

            // Determine the max power + refresh
            pout = adc_iout * adc_vout;
            if (pout &gt; MPP_power){
                MPP_power = pout;
                MPP_Duty = duty;
            }
        }
        //Add code to update EPWM register
    }
}</pre>
```

Mode 0: flag_sweep = 0, duty is a fixed value.

Vinput of ADC

Iinput of ADC.

Mode 1

MPPT Algorithm.

Lab Results

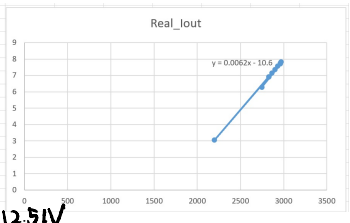
1. Provide a snippet of your code showing the calculation of the voltage at the input of the ADC pin. Provide a screenshot of the CCS debugger showing the calculated voltage.

2. Provide a screenshot of your MPPT algorithm implemented in code.

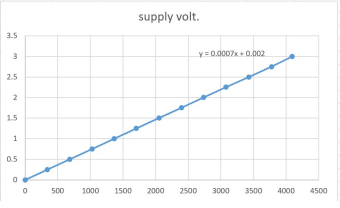
3. Provide the maximum power point identified by your control for both full and half I_{SC}. For each point, provide the duty cycle, input voltage, and output power as measured by your microcontroller.

(1). screenshot of Vout and Iout (True value and calculated values)

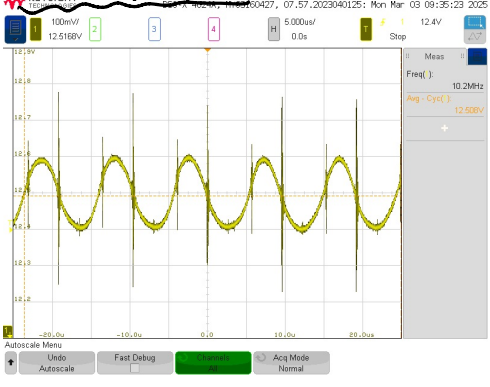
ADC_raw_iout	Real_Iout
1690	0
2830	6.916
2865	7.136
2900	7.359
2930	7.573
2960	7.746
2970	7.82
2200	3.05
2960	7.7
2747	6.276
2830	6.9158



ADCRESULT0	supply volt.
0	0
340	0.25
685	0.5
1025	0.75
1365	1
1701	1.25
2050	1.5
2395	1.75
2735	2
3080	2.25
3430	2.5
3775	2.75
4095	3



Actual Vout = 12.51V



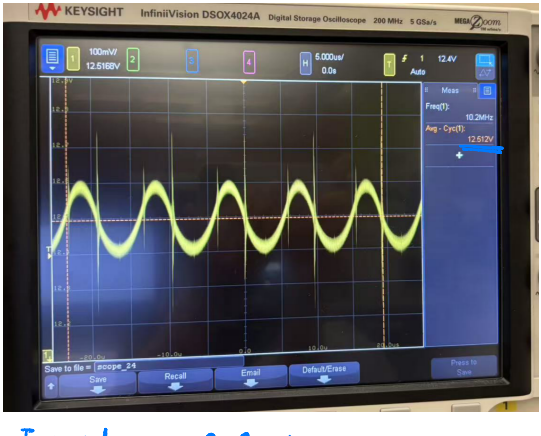
estimated Vout = 12.36V

within 5% ✓

MPPT_ duty ≈ 61.5%

duty_cmp	unsigned int	295	0x0000A9
duty	float	0.615000069	0x0000A9
adc_raw_vout	unsigned int	1537	0x0000A9
adc_vout	float	12.3860807	0x0000A9
adc_raw_iout	unsigned int	3006	0x0000A9
adc_iout	float	7.9041338	0x0000A9
MPP_Duty	float	0.615000069	0x0000A9
MPP_power	float	97.3910446	0x0000A9
pout	float	97.8375397	0x0000A9
flag_sweep	int	1	0x0000A9

(3). Full I_{SC} = 5.21 A.

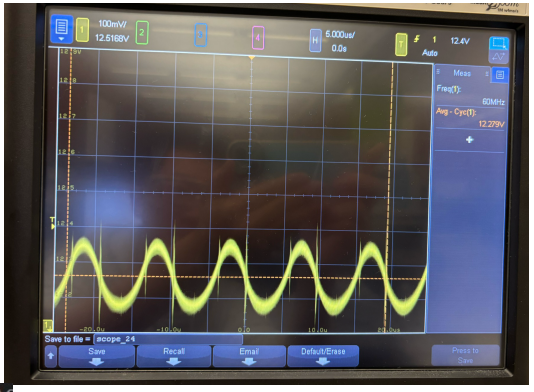


Estimated power: 97.83W

Actual power: 7.9674 × 12.512 ≈ 99.6W

& 95.61W

between 99.6 & 95.61W



duty_cmp	unsigned int	297	0x0000A9
duty	float	0.619999945	0x0000A9
adc_raw_vout	unsigned int	1510	0x0000A9
adc_vout	float	12.1684971	0x0000A9
adc_raw_iout	unsigned int	2346	0x0000A9
adc_iout	float	3.81939411	0x0000A9
MPP_Duty	float	0.619999945	0x0000A9
MPP_power	float	46.4762878	0x0000A9
pout	float	46.4762878	0x0000A9
flag_sweep	int	1	0x0000A9

when $I_{sc} \approx 2.60A$, $V_{out} \approx 12.28V$, $I_{out} \approx 3.95A$.

$\Rightarrow$ Estimated power $\simeq$ 46.47W

Actual power = 47.47W.

basically because at low current, the value is not so correct.