Name:
Time – Pre Lab:
Time – Lab Exercises:

Maximum Power Point Tracking for Solar Arrays

Pre Lab

Bring the I-V curves you took in Lab 1 with you to lab (soft copies are fine).

Concept of maximum power point tracking

1. What is maximum power point tracking and why is it necessary for solar arrays?

2. Name a commonly used control algorithm for solar maximum power point tracking.

Choppers for MPPT: Almost all solar MPPTs use a chopper of some sort. We will be using a boost chopper to implement one in lab, but the basic algorithm for using a chopper as a MPPT is the same for all types.

1. Explain the **basic** steps for using a chopper as a maximum power point tracker for PV with the algorithm you listed above. Don't list more than 4 or 5 steps. Use a block diagram is that's easier for you.

Name:
Time – Pre Lab:
Time – Lab Exercises:

Control Systems:

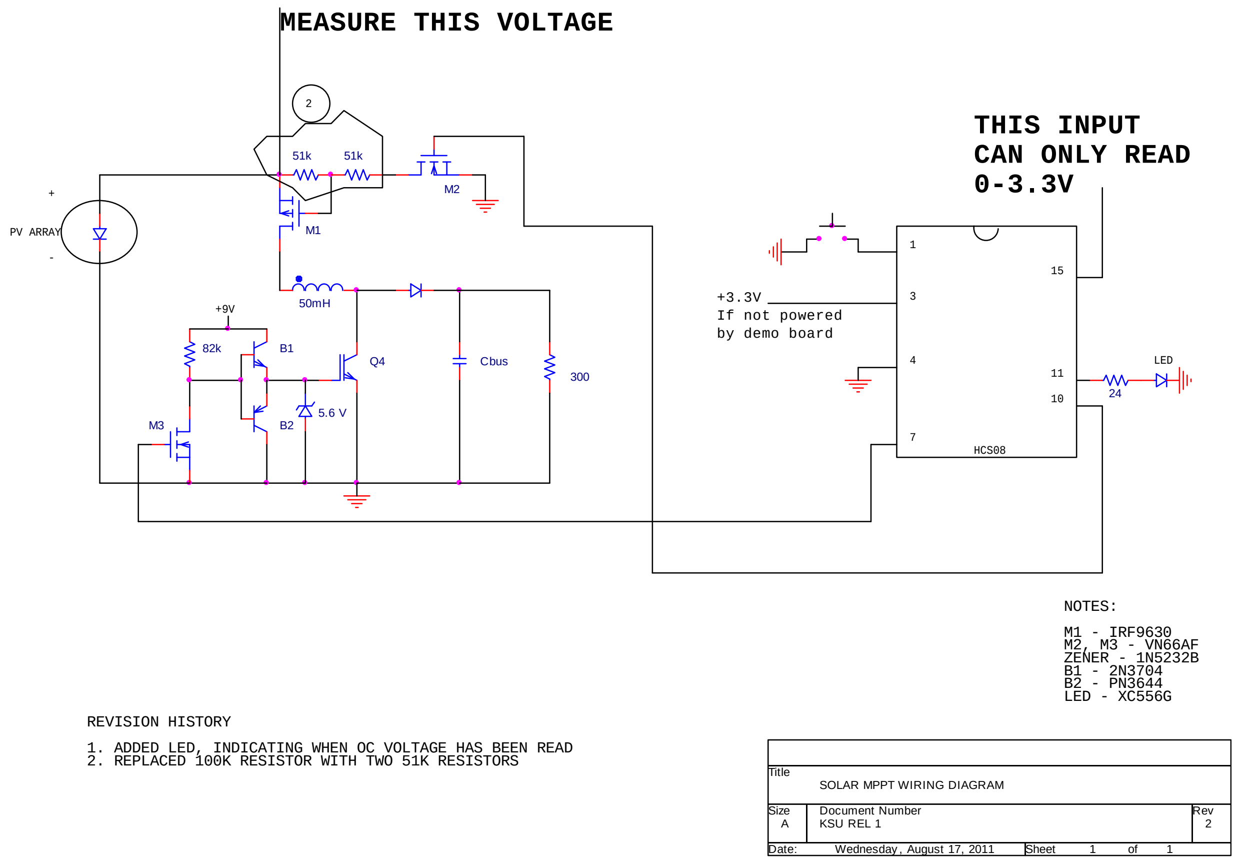
1. Read the section of this lab on control theory that appears at the beginning of Part II of the Lab Exercises. Write down any questions you have and bring them to lab.

Lab Exercises

You are going to implement a MPPT that uses a constant voltage ratio algorithm.

There are two parts to this lab: an electronics part and a control systems part. You have been provided with the bulk of both the hardware design and the software for the maximum power point tracker and controller. You are free to deviate from and/or change anything you've been given, but be aware that the more you change the less the lab instructor will be able to help you. No matter what you will still be expected to produce a working maximum power point tracker by the end of the lab.

Part I - Electronics: The following circuit is most of a solar MPPT. You've been given HCS08 C code to use for the microcontroller (uC).



Circuit 1 Partial MPPT

In order to implement the constant voltage ratio algorithm, we must be able to measure the voltage across the array both when it's open-circuited and when it's connected to the rest of the circuit. The A/D converter of the HCS08 microcontroller can read a voltage, but it must be between 0V and the voltage applied to its Vdd pin (3.3V).

HCS08 note: During the course of this lab it's probably easiest to use the HCS08 demo board to debug your code. While you're using the demo board, don't connect 3.3V to pin 3 (Vdd) of the HCS08. The HCS08 will be powered by the demo board. The pins of the HCS08 may be accessed via the I/O port connector (the black terminals on the bottom of the demo board). See the document " I O port connector" for a map of how those terminals correspond with the pins of the HCS08.

1. Build Circuit 1. Use the circuit-testing HCS08 code provided to verify proper operation. If the circuit is built and connected correctly the chopper should run at 500Hz with 15% duty cycle. Pushing the button should toggle the PV panel in and out of the circuit. The LED should flash once when the PV array is connected to the circuit.
2. From this point on use the code "MPPT controller." Design a circuit to interface the array voltage with the HCS08 so that whenever the software wants to, it can read the voltage across the array. The software uses the actual PV voltage in its calculations, so you must factor any scaling you do into the software. You'll find the scaling code clearly marked in the code.
3. To test whether or not your circuit works run the chopper with the LabVolt software, rather than your circuit, controlling Q4. That means don't hook anything up to the gate of Q4. When you push the button in your circuit, the uC will open-circuit the array and read the voltage. That voltage will be stored as Voc in the uC memory. The uC will also read the voltage periodically with the PV array in the circuit. That voltage is stored in the uC memory as v. Both of those variables should consistently be very close to what you measure with a meter.
4. When your circuit is working, draw a schematic of it and turn it in with the lab.

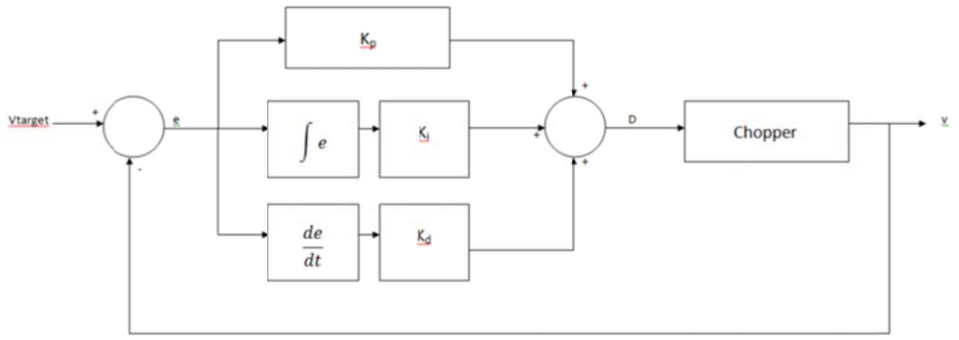
<u>Part II – Control Systems</u>

When Part I is working the software will get the PV voltage whenever it needs it, be that open-circuit or in-circuit. For Part II you are tasked with writing the code that will make the in-circuit voltage what it needs to be. The software will give you a target voltage ("vtarget") based on what you enter for the variable "ratio," which you found in Lab 1. The software will measure the in-circuit voltage and will allow you to change the duty cycle of Q4 every 80 ms. By changing the duty cycle you can change the array voltage until that voltage is close to "vtarget."

One common way to make the voltage what you need it to be is to use a PI or PID control scheme. For those of you who haven't used a controller like this, here's a crash course:

We often represent control systems with block diagrams. Our block diagram would look like this if we used a PID controller:

Name:
Time – Pre Lab:
Time – Lab Exercises:



The Chopper block represents the boost chopper, as you may have guessed. The boost chopper has a certain transfer function that describes how the PV voltage changes if the duty cycle (D) changes. We're not going to worry about what that transfer function is for. All we care about is that the chopper takes a duty cycle and it gives us a voltage.

In order to make the chopper work, we must give it a duty cycle. That's where the PID controller comes in. The PID controller uses the difference between vtarget and v, which we call error (e), to determine what duty cycle to give to the chopper. That relationship can be seen with these two equations:

$$e = vtarget - v \tag{1}$$

$$D = e * k_p + \int e * k_i + \frac{de}{dt} * k_d + D_0 \tag{2}$$

Equation 1 is easy and self explanatory. Equation 2 has a little more to it. Without getting into too much control theory, here's what each term means in equation 2:

- k_p is called proportional gain. If we only used a proportional (P) controller, D would always be directly proportional to the error between vtarget and v. P controllers generally aren't very stable and they can never make the error go to zero, which is the goal.
- k_i is called integral gain. By taking the integral of the error, our controller gains some knowledge of what the gain has done in the past. It looks backward, so to speak. When we implement integral control discreetly, as we do with a uC, the integral becomes a sum. Integral control is often combined with proportional to make PI controllers. PI controllers can be very stable and can make the error go to zero.
- k_d is called derivative gain. By taking the derivative of the error, our controllers can get some idea of where the error is going in the future. It looks forward, so to speak. Derivative control is often combined with both proportional and integral to make PID controllers. When we implement derivative control discreetly, as we do with a uC, the derivative becomes a difference. Sometimes it's also used with just proportional to make PD controllers. Like PI, these controllers can be very stable and can make the error go to zero.
- D_0 is the output of the controller if all of the other terms are zero. This term is not represented in the block diagram. It's a very practical term to have though as it helps to stabilize the system

on startup. You are free to set D_0 to whatever value you like. You'll find it clearly marked in the code.

If you choose to use one of those schemes be aware that the transfer function of a boost chopper has a negative in front of it when it's used as a MPPT. To compensate for that you can use what's called "direct action." All that means is that you make the gain terms negative.

When controllers don't work, the system they're supposed to be controlling goes unstable. In our case what this means is the duty cycle will go to 0 or 100% or it will bounce back and forth between the two. Obviously if the duty cycle is 100% that means the array is shorted. While this condition isn't ideal, it's not going to kill our PV panels either. Keep a close eye on the voltage across the array as you're testing your controller and turn your circuit off or disconnect the PV array if you see the array voltage drop to near 0V. With that said, it's perfectly acceptable to start with initial guesses for the gain terms and run the MPPT. Chances are it will be unstable, so you can then make changes to the gain terms and try again…and again…and again…

1. Enter your OC/MP voltage ratio into the code where it's called for (the variable "ratio").

2. Design and implement a control system for the array voltage.

3. Demonstrate your working MPPT to your lab instructor.
4. Copy and paste your code into this document.