NC STATE UNIVERSITY

Buck Converter and Constant-Current Driver

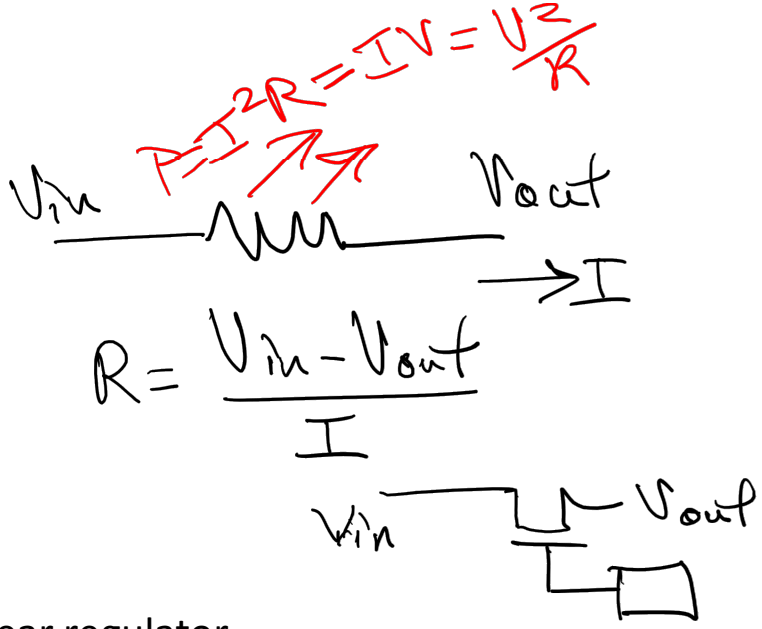
v3

A.G.Dean
ECE 460/560
Embedded System Architectures

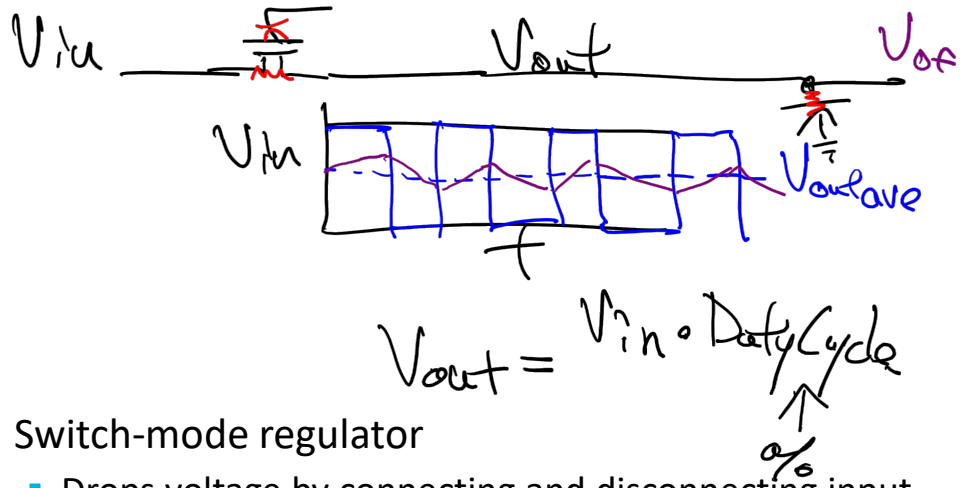
NC STATE UNIVERSITY

Switch-Mode Power Conversion and the Buck Converter

How to Lower a Voltage from V_{in} to V_{out}



- Linear regulator
  - Drops input voltage using "pass" transistor as variable resistor, converting extra power to heat.
  - If output doesn't need as much power, reduce conductance of transistor (raise resistance), wasting more power.



- Switch-mode regulator
  - Drops voltage by connecting and disconnecting input voltage very frequently
  - If output doesn't need as much power, reduce duty cycle (fraction of time) input is connected.
  - Minor additional power loss.
    - Switches (transistors, diodes) are either on (very low resistance) or off (open circuit)
    - Duty cycle typically controlled by pulse-width modulated (PWM) signal
  - If needed, filter output voltage with capacitor

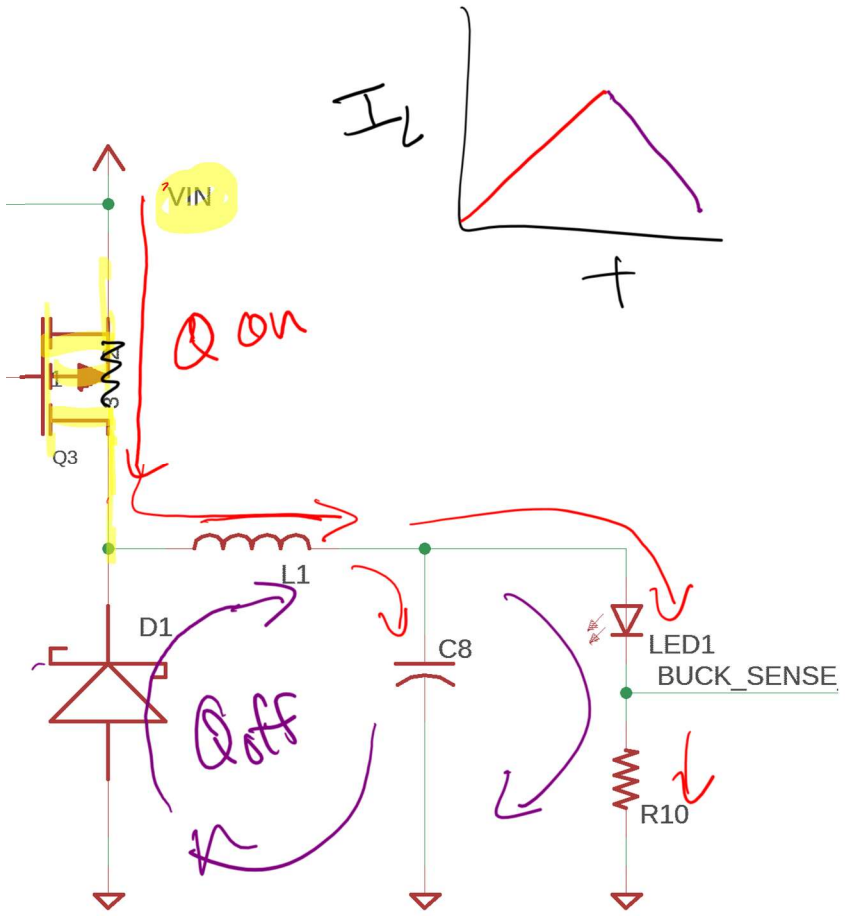
NC STATE UNIVERSITY

Energy Storage and Filtering

- Convert energy between electrical and/or magnetic forms for efficient storage and filtering
- Use capacitors and inductors to store energy and filter out ripple
  - Capacitors – **voltage** is stabilized
  - Inductors – **current** is stabilized
- Capacitor stores energy as voltage difference
  - $E = ½ CV^2$
  - Total energy α voltage^2: capacitor needs higher voltage rating
  - Current affects charge/discharge rate, but not total energy storage
- Inductor stores energy as current and its electromagnetic field
  - $E = ½ LI^2$
  - Total storage α current^2: inductor needs higher current rating
  - Voltage affects charge/discharge rate, but not total energy storage
  - Good match for LED driver application. Easier to increase current capacity.

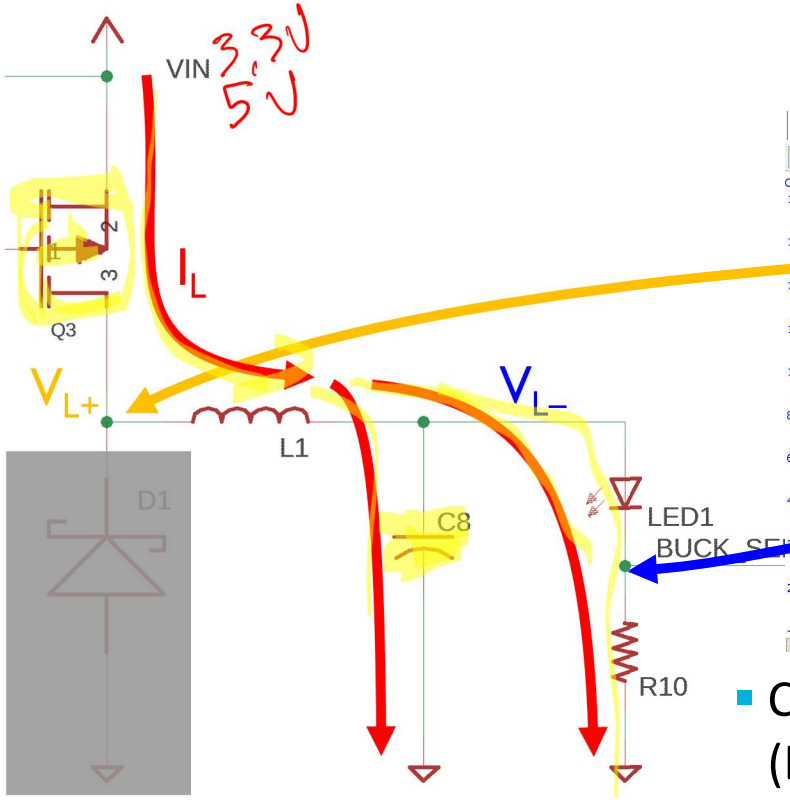
NC STATE UNIVERSITY

Buck Converter Overview

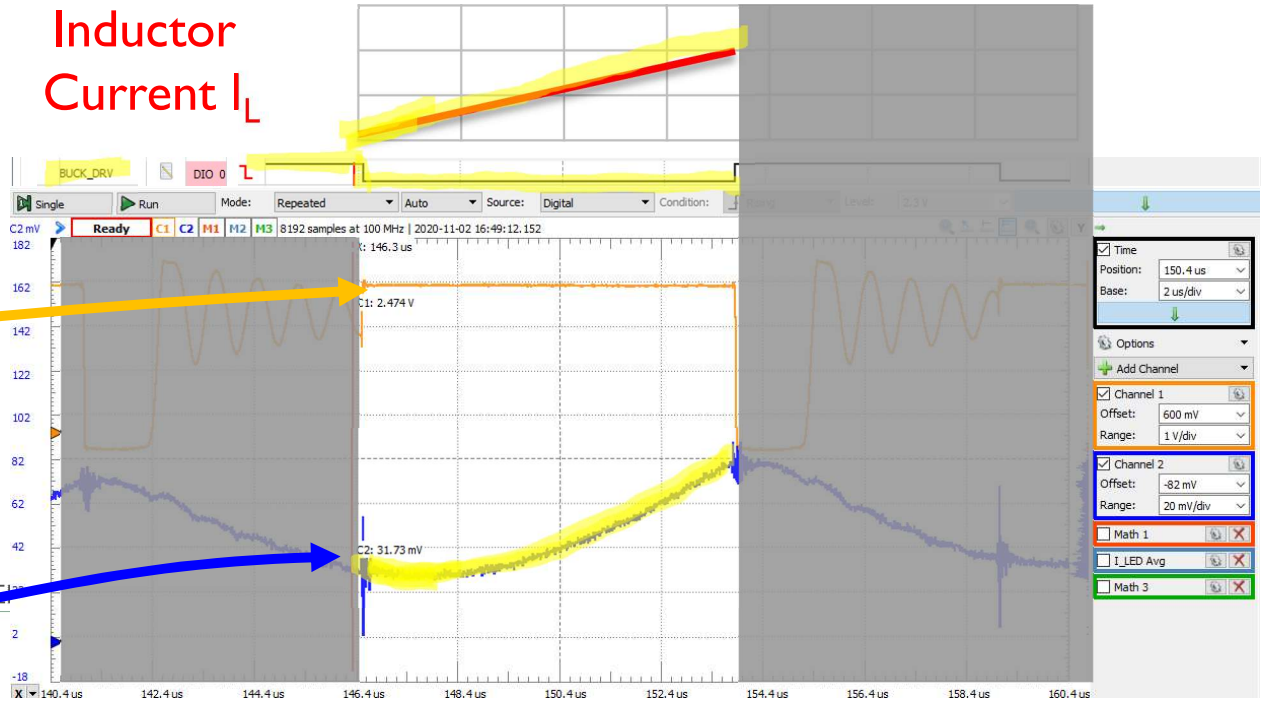


- PWM input signal
  - Duty cycle determines control effort
- Switches
  - Transistor Q3 is on when PWM signal on gate is active (low)
  - Diode D1 is off when Q3 is on (controlled by voltage at Q/D/L node)
- Can add capacitor C8 on output to store some energy

Switch Q3 Closed

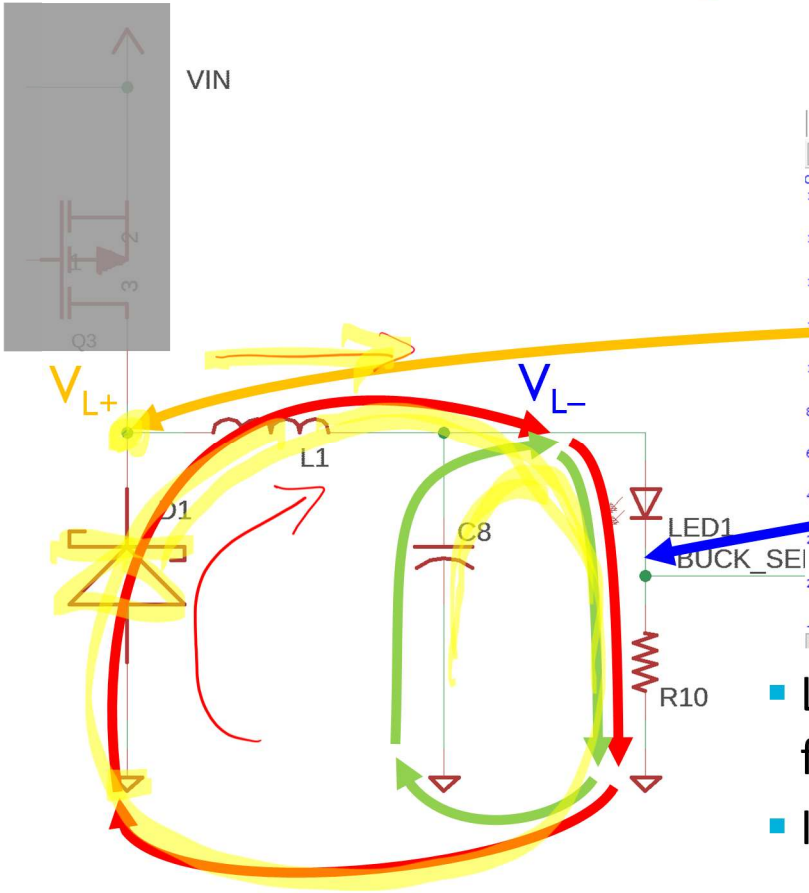


Inductor Current I_L

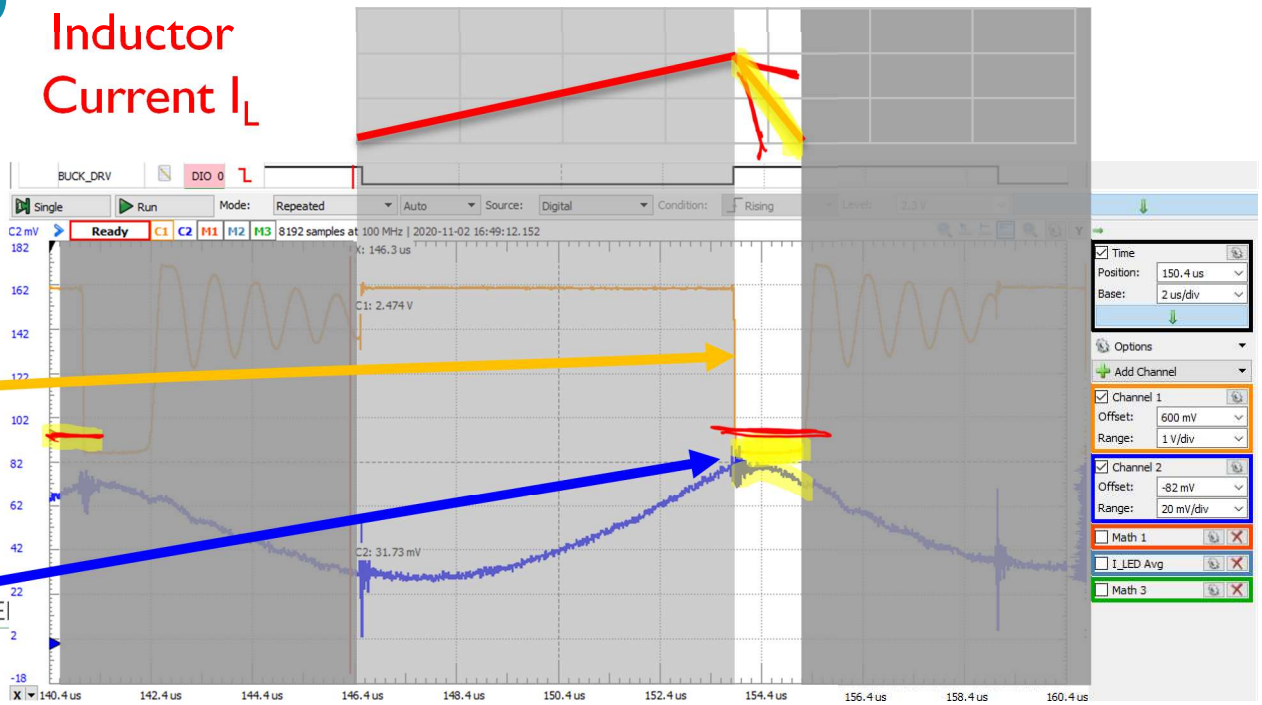


- Current flows through inductor L1 (ramping up) to load (LED1, C8)
  - $di(t)/dt = (V_{L+}-V_{L-})/L$
  - Slope depends on voltage applied across inductor
- Charges up L1 (inductor creates magnetic field) and C8

Switch Q3 Open, I_L>0

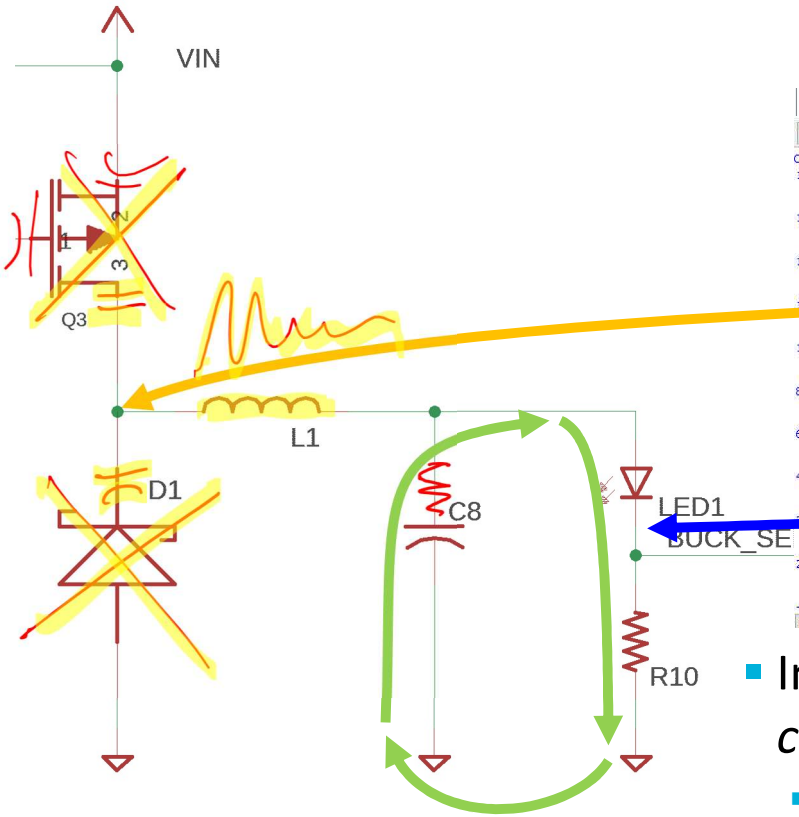


Inductor Current I_L

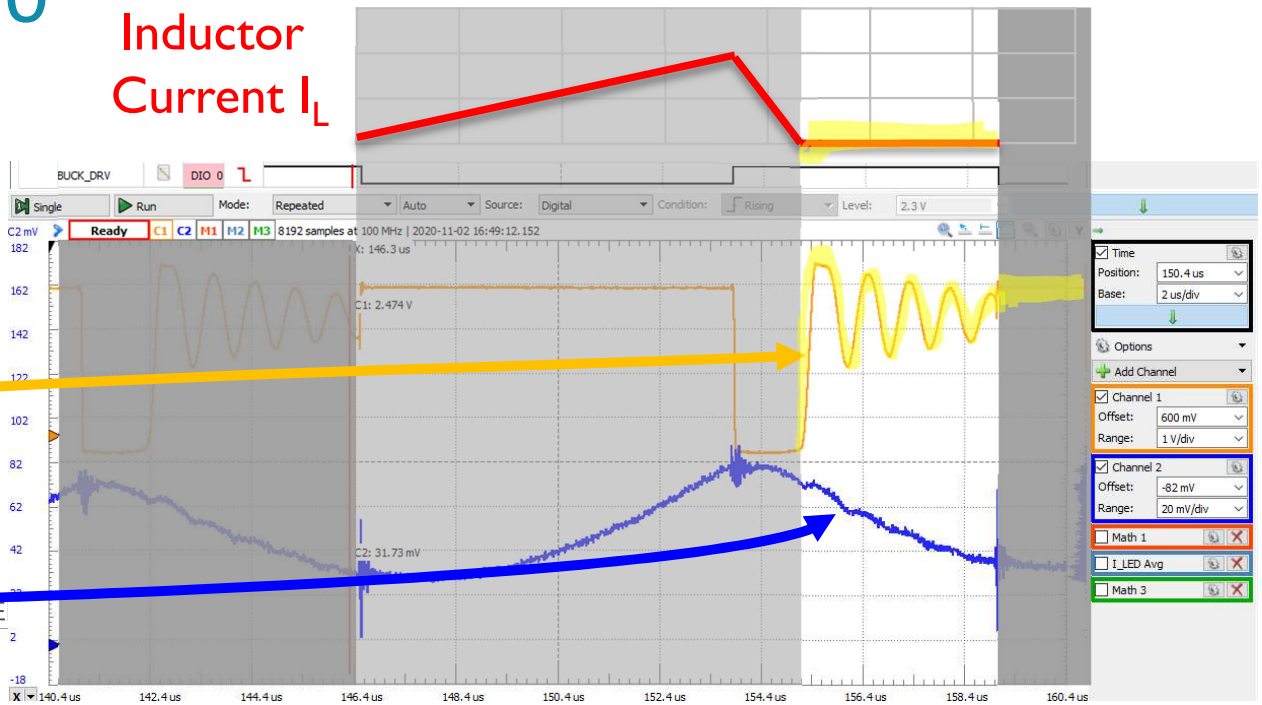


- Load (LED) is powered by current from L1 and C8 (for filtering
- Inductor current flows through D1 now, ramps down
  - Larger load current will make I_L ramp down faster

Switch Q3 Open, $I_L = 0$

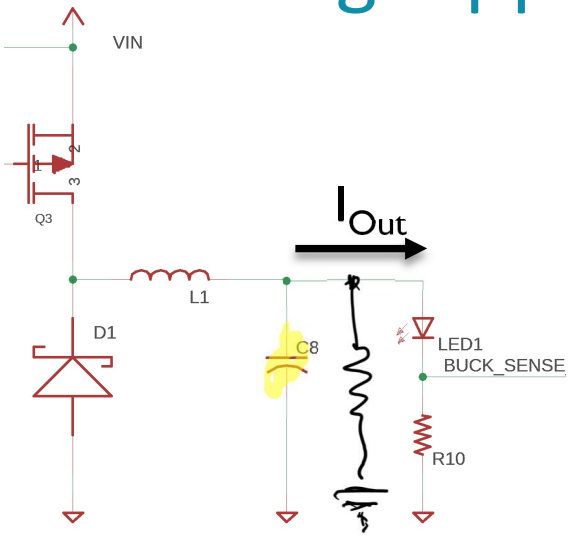


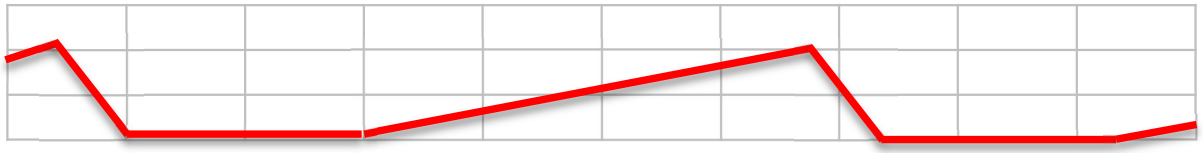
Inductor Current I_L

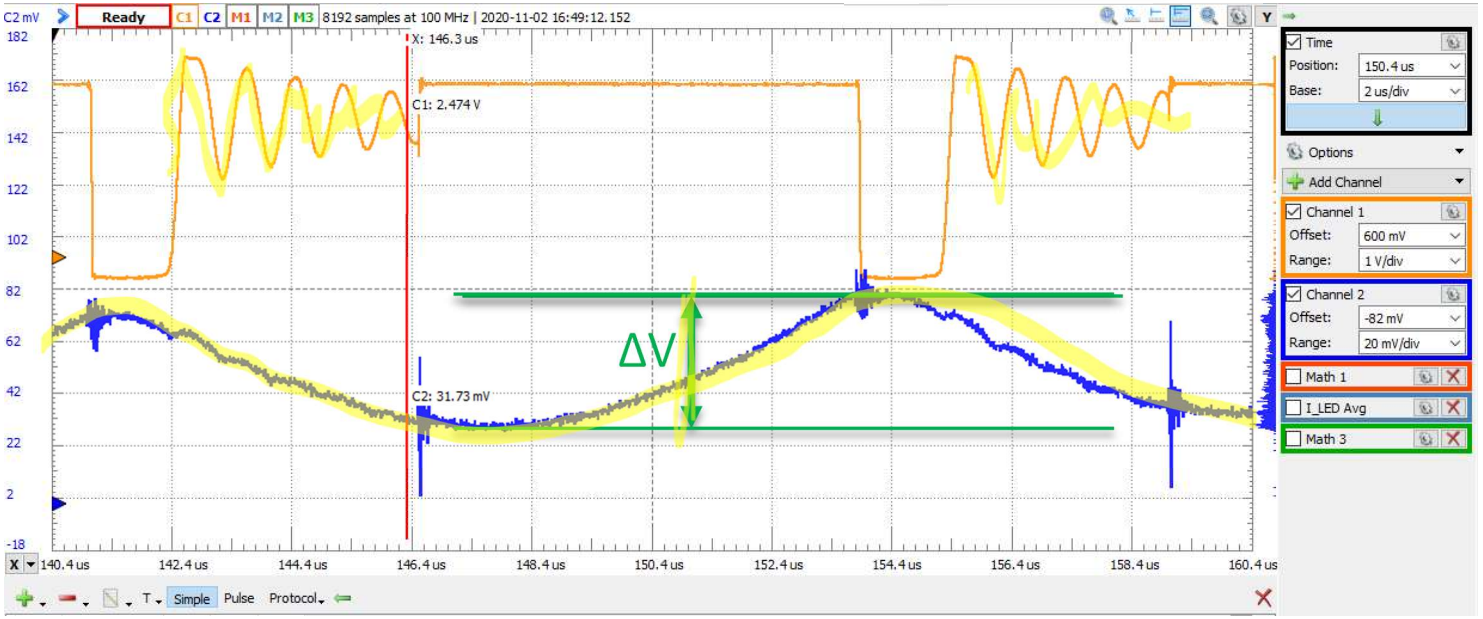


- Inductor current reaches 0, so now in *discontinuous conduction mode*
  - Models (equations) get more complex
- Load is powered by C8
- Oscillation from L1 and parasitic capacitances of D1 and Q3

Switching Ripple







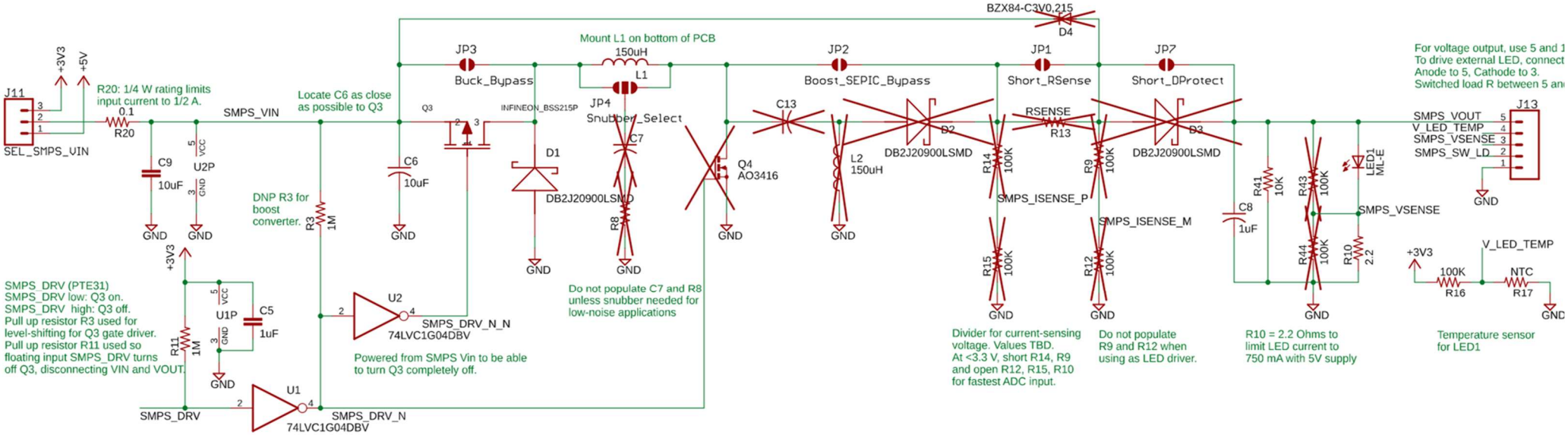
- Switching creates ripple in I_L,V_{Out}
- Output ripple voltage
  - Depends on load current I_{Out}, C8, switching frequency
  - $\Delta V = \frac{I_{Out}}{4Cf}$

NC STATE UNIVERSITY

Expansion Shield's Switch-Mode Power Converter

NC STATE UNIVERSITY

Shield Switch-Mode Power Converter Overview

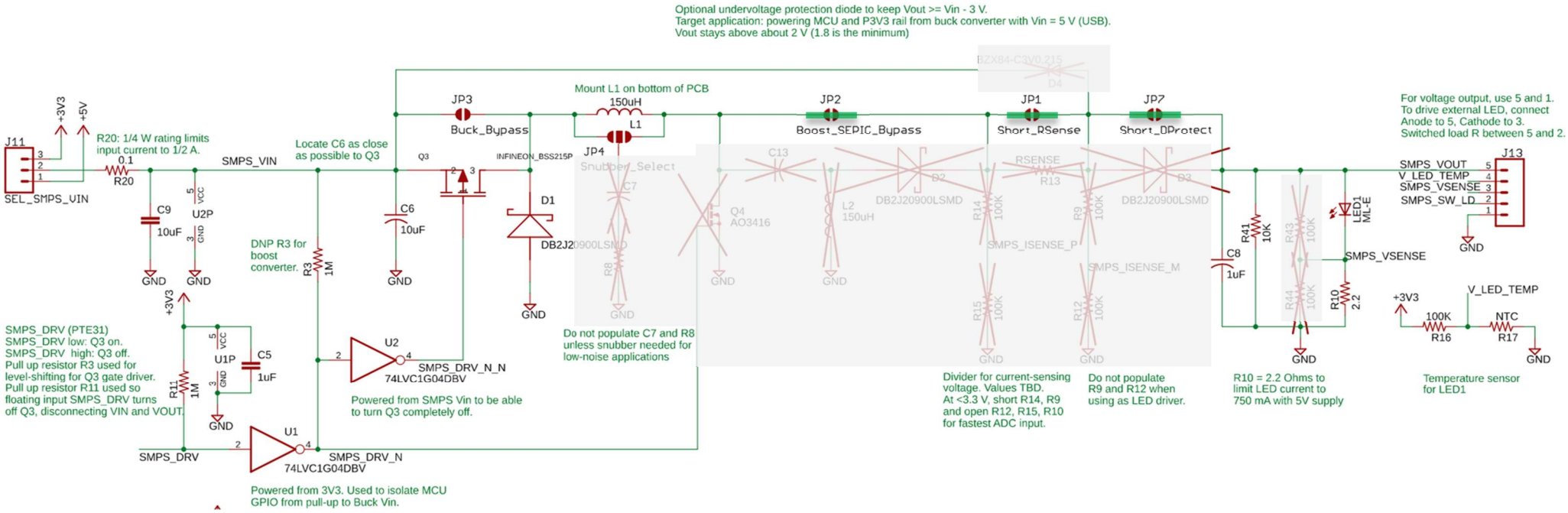


- PCB supports different converter types
  - Buck converter (reduce voltage) – we use this
  - Boost converter (increase voltage)
  - SEPIC (raise or lower voltage, or neither)
- Noise control
  - Snubber (R8, C7) to reduce EMI when Q3 switches on, off
- Switchable load transistor (Q2)
  - For testing response to changes in load

- Feedback options
  - Output voltage scaled by R9/(R9+R12). Single-ended (ground-referenced)
  - Current out **through LED1**
    - Single-ended (ground-referenced)
    - Scaled by R10 (2.2 mV/mA)
    - R10 selected to limit maximum current through LED1 to safer level
  - Current out **before capacitor C8**,.
    - Differential (not ground-referenced)
    - Scaled by sense resistor R13

NC STATE UNIVERSITY

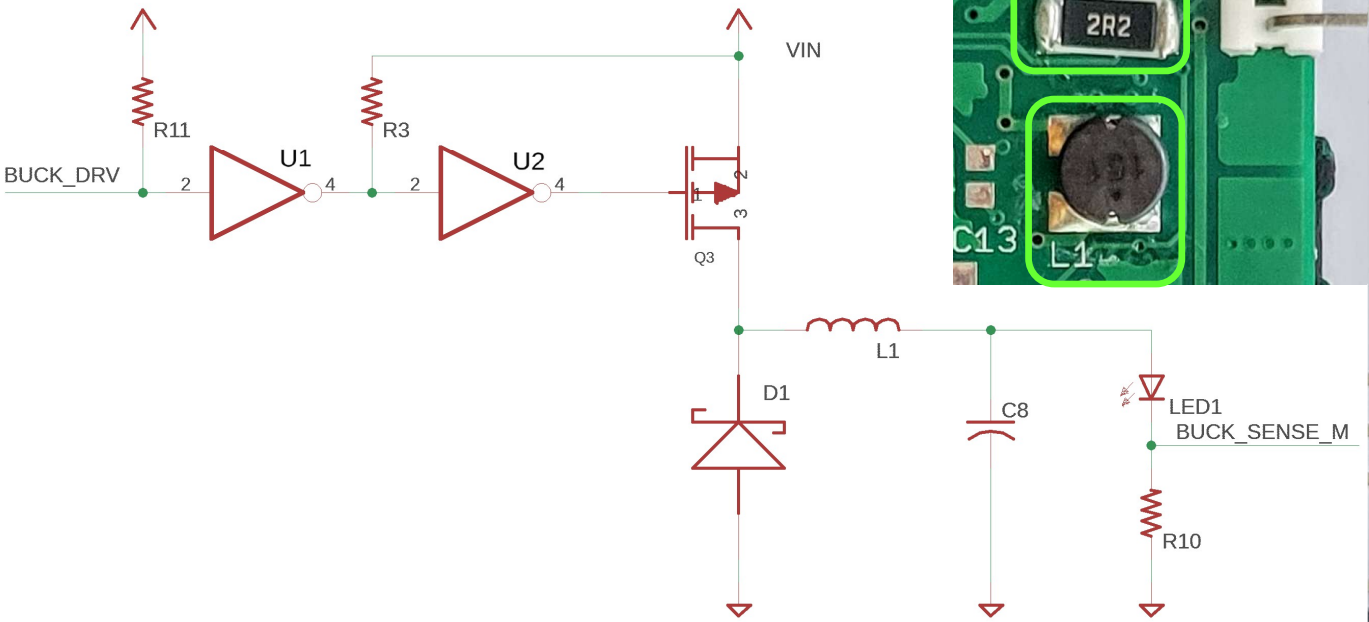
SMPS Configured as Buck Converter

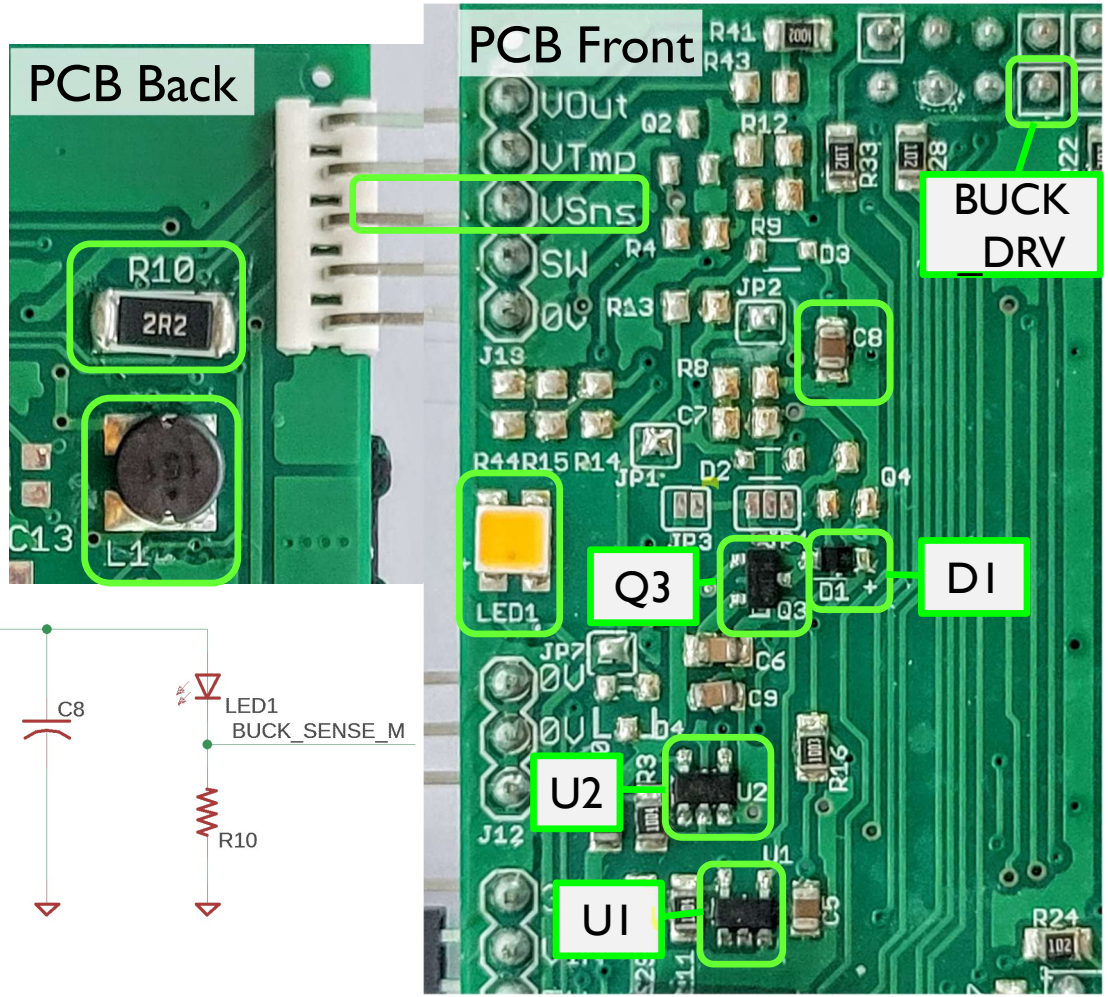


- Unused components are grayed or X'd out
- Shorted solder jumper

NC STATE UNIVERSITY

Buck Converter on PCB



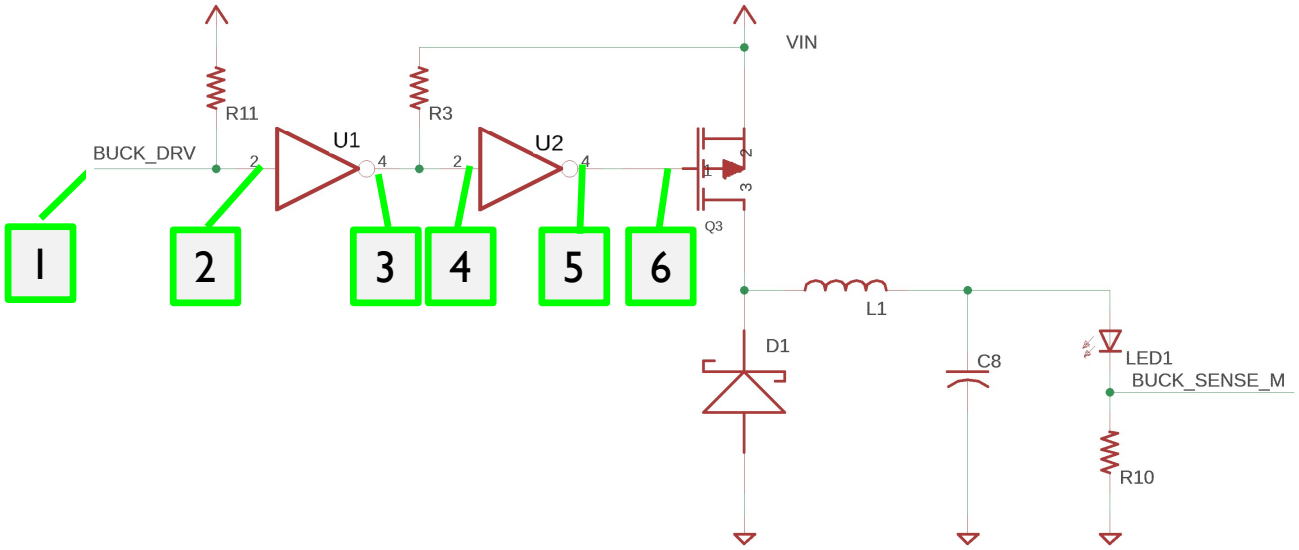


NC STATE UNIVERSITY

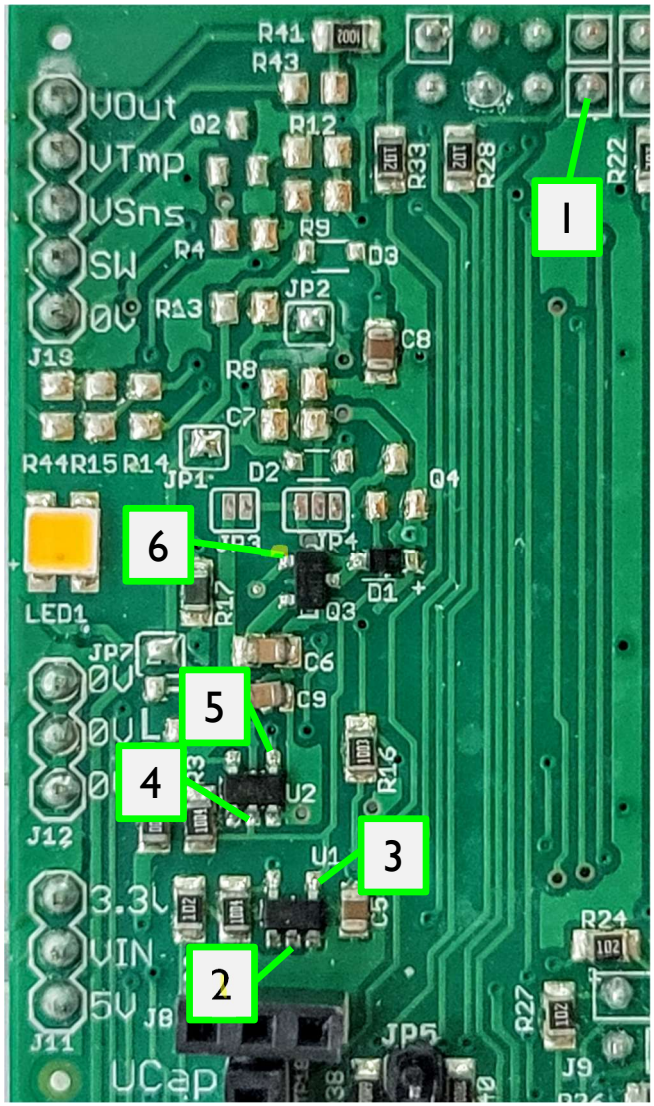
Deep Dive into Circuit and PCB (Optional)

NC STATE UNIVERSITY

Tracing the BUCK_DRV PWM Signal

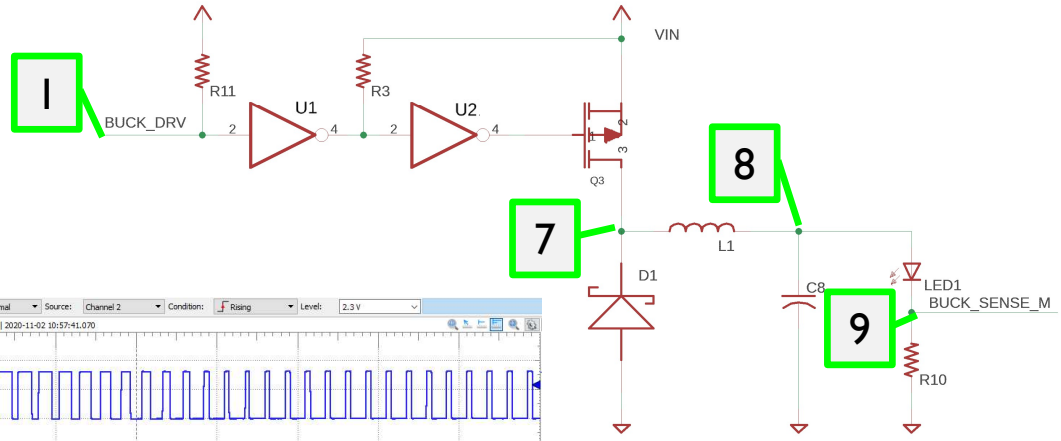


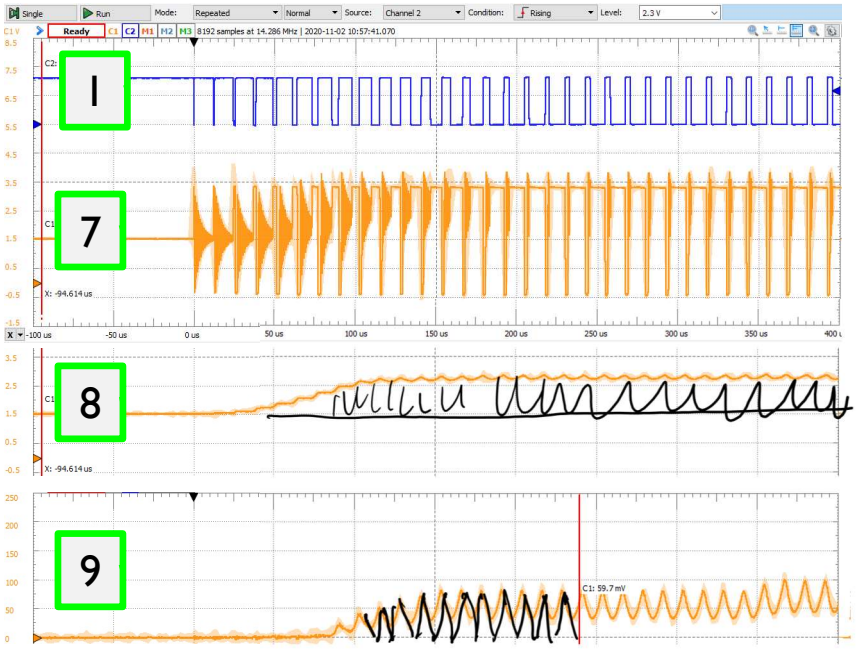
1. Confirm PWM signal at header pin
2. Confirm PWM signal at input to U1
3. Confirm inverted PWM signal at output of U1
4. Confirm inverted PWM signal at input to U2
5. Confirm PWM signal at output of U2
6. Confirm PWM signal at gate of Q3

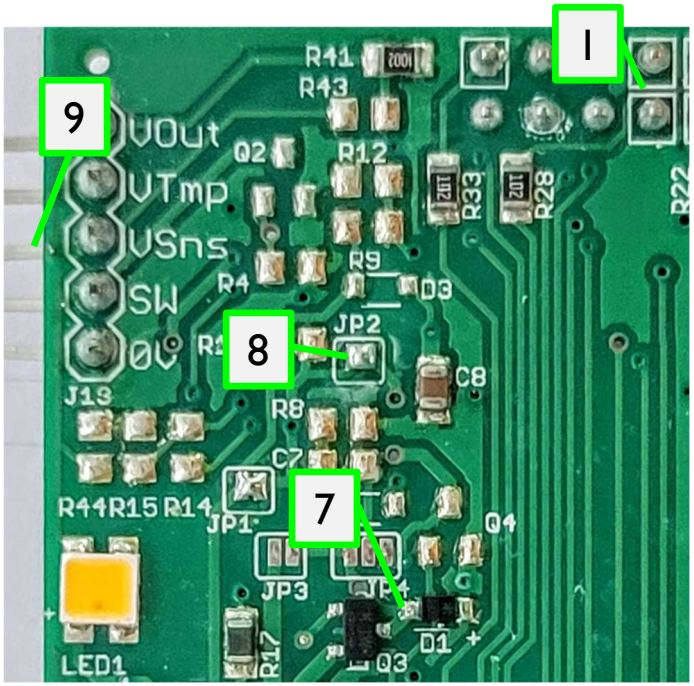


NC STATE UNIVERSITY

Buck Converter Power Stage Signals



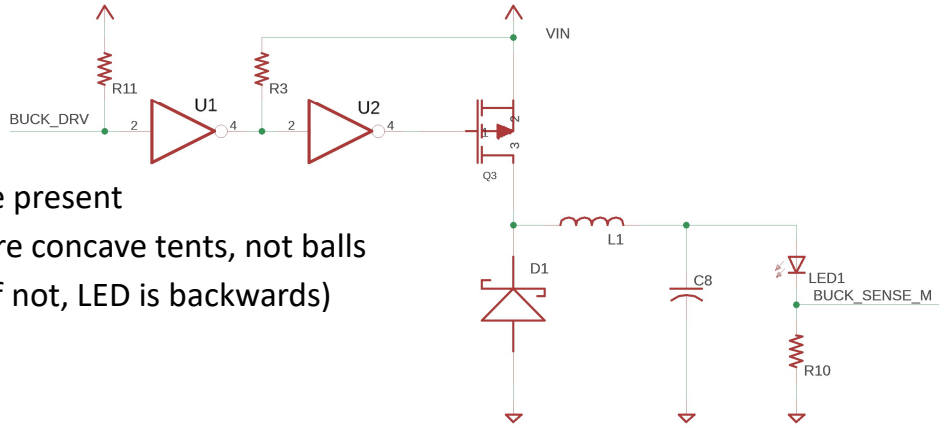


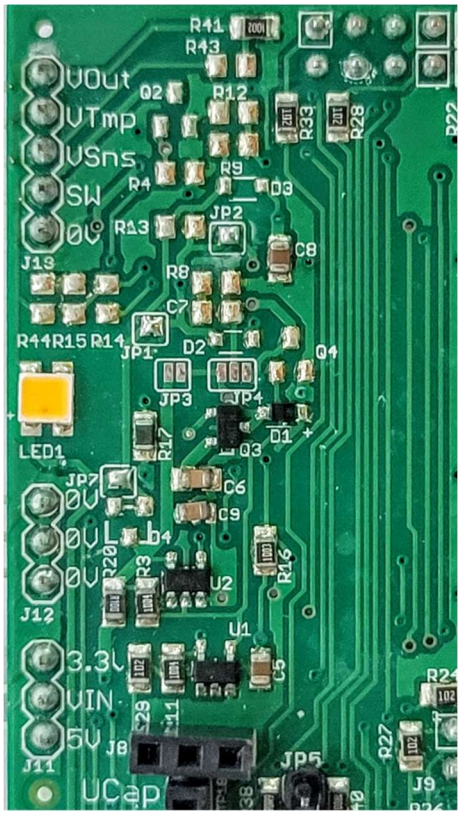


C8 No. C8

7. Confirm switching signal at Q3/D1/L1 node. 1 V/division
8. Confirm output voltage at JP2 (L1/C8/LED1 node). 1V/division. *As C8 not used, will have large ripple.*
9. Confirm current sense voltage at BUCK_SENSE_M (S-). 50 mV/division. *As C8 not used, will have large ripple.*

Visual, Resistance and Diode Tests – POWER OFF!



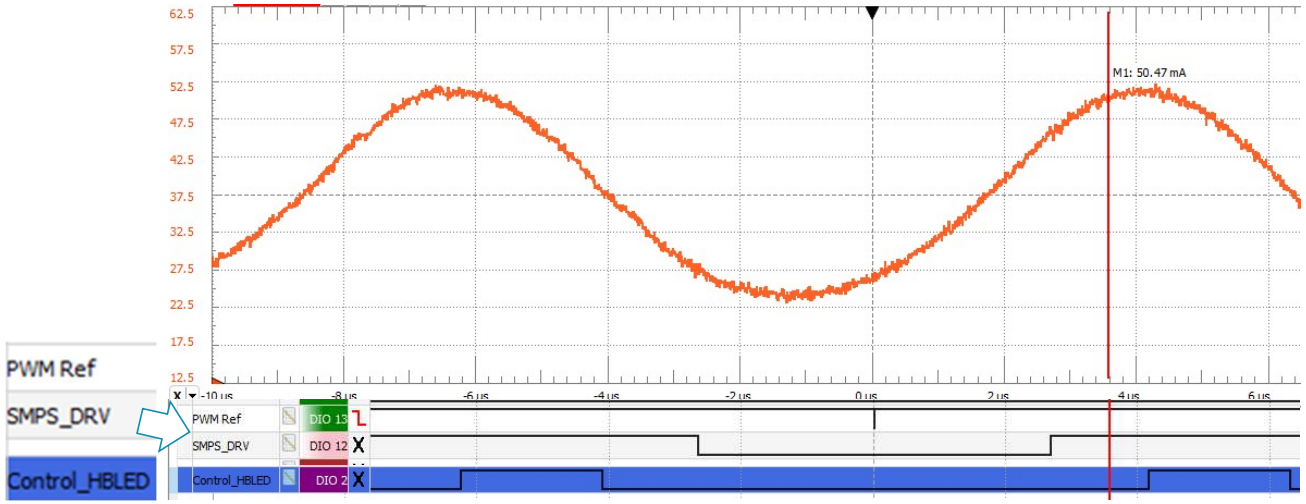


- Remove power and J11 jumper
- Visual checks. Confirm:
  - All components in this schematic are present
  - Solder connections to header pins are concave tents, not balls
  - Dot on LED1 (anode) is closer to + (if not, LED is backwards)
  - JP1 and JP2 are shorted
- Multimeter resistance tests
  - J13 S- to 0V should be 2-3 Ω. If larger, check for bad soldering on R10, J13
  - BUCK_VIN to ground should be large (>1M, rising)
  - Q3 pin 3 / D1 Anode / L1 node to ground should be large (>1M, rising)
  - JP1 to ground should be large (>1M, rising)
- Multimeter diode tests
  - D1 should show forward voltage of about 0.24 V
  - LED1 should show forward voltage of about 2.46 V
  - Q3
    - + on Gate (2), - on Drain (3) forward voltage 1 V
    - + on Drain (3), - on Gate (2) forward voltage 0.24 V

NC STATE UNIVERSITY

Architecture for Constant Current HBLED Driver using Buck Converter

Timing Questions – How does it Behave?

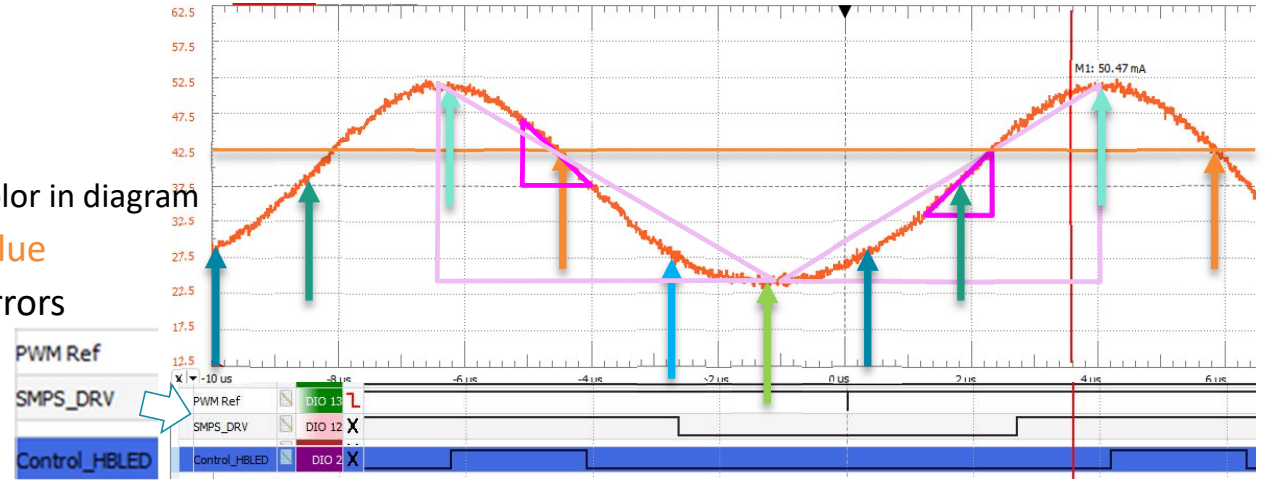


- How tight are the input and output timing requirements for this application?
  - How large can timing jitter for sampling be and still get good enough performance? 100 us? 10 us? 1 us? 0.1 us?
- Use mixed-signal scope display
  - Shows analog and digital signals on same timeline
  - Can show us *what* and *when* the system *really* does things
  - Analyze to see timing relationships and details of system components in processing chain
- Example
  - Orange waveform is LED current in mA
  - PWM Ref is timing reference output from Timer
  - SMPS_DRV is MCU output to switch transistor on (0), off (1)
  - Control_HBLED is "twiddle bit" indicating function execution
- How long does it take to run the whole control loop?
  - Duration limits maximum control loop frequency
  - How consistent is this execution time?
  - How long does each individual part take?
    - Control_HBLED is the main part, but there are others

NC STATE UNIVERSITY

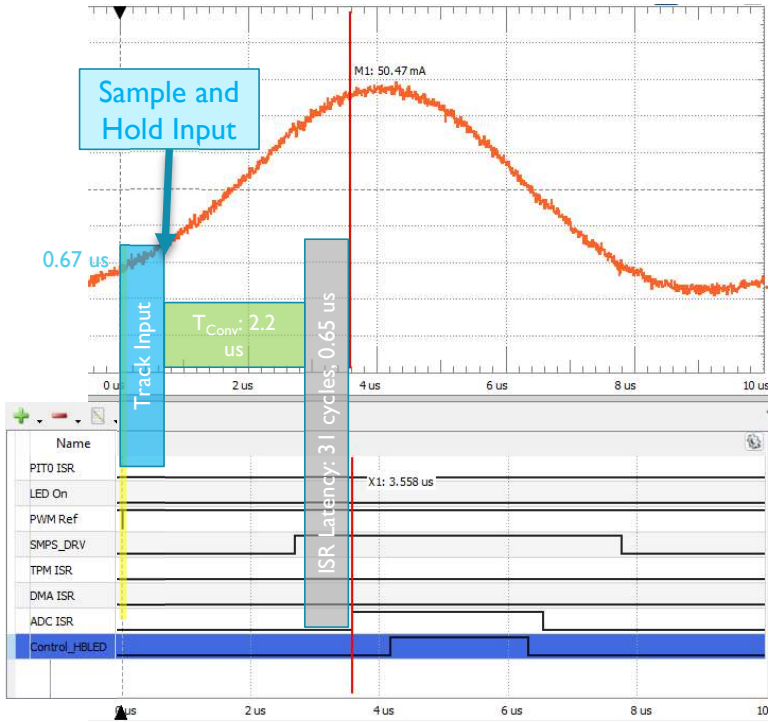
Timing Questions – When Should We Sample?

- Sample input signal synchronously
  - Same phase every time = arrows with same color in diagram
- Ideal time to sample: near average signal value
- Signal slope indicates sensitivity to timing errors
  - $= \frac{change\ in\ current}{change\ in\ time}$
- Rough estimate
  - Absolute: $(I_{max} - I_{min})/(T_{max} - T_{min})$
    - (50 mA – 25 mA)/(4 us – –1 us) = 5 mA/us
  - Relative, based on average current:
    - ((50 mA – 25 mA)/37.5 mA)/5 us = 13%/us
- Note that slope depends on signal phase
  - ~0 at peak and trough
  - Larger slopes elsewhere, especially near midpoint
  - Slope varies more when falling than rising, since dependent on load current



- Estimate maximum sensitivity
  - Absolute: (10 mA)/(1.1 us) = 9 mA/us
  - Relative: (10 mA/37.5 mA)/1.1 us = 24%/us
  - Actual sensitivity depends on phase of sampling
- Are there any hardware signals available at/near that ideal time, simplifying sampling sync.?
  - Trigger A/D conversion with PWMRef?
  - PWM rising edge? Falling edge?
    - What about switching noise?

When does ADC ISR Start, and Why?



- ADC triggered by PWM Ref signal
- ADC ISR starts 3.558 us later. Where does the time go?
- Evaluate timing of A/D converter operation
  - See MCU reference manual's ADC chapter
  - Track input: 0.67 us, adjustable
  - Sample and hold at end of tracking
  - Convert: 2.2 us
- ISR has 15 cycle latency
  - 15/48 us = 0.313 us
- Total: 3.183 us
- Debug signal starts at 3.558 us
  - 3.558 us – 3.183 us = 0.375 us late.
  - 0.375 us * 48 MHz = 18 cycles
- Why is it late?

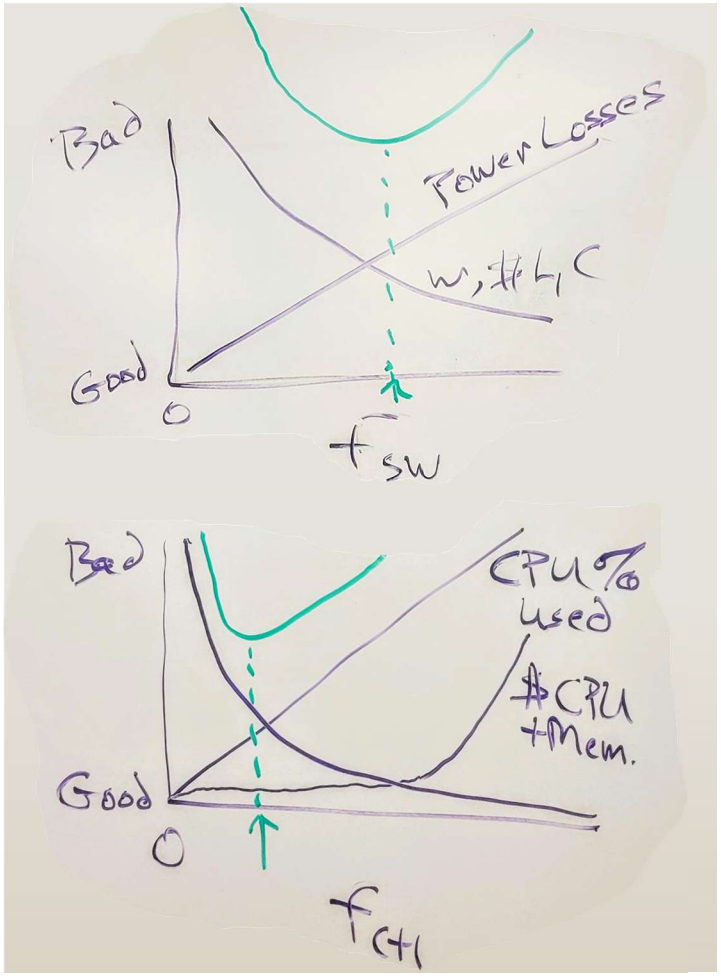
```
ADC0_IRQHandler() {
5       PUSH    {r4-r5,r7,lr}
n/a   DEBUG_START(DBG_ADC_ISR);
2       LDR     r0,[pc,#384]
2       LDR     r0,[r0,#0x10]
1       MOVS    r4,#0x01
1       LSLS    r4,r4,r0
2       LDR     r0,[pc,#380]
2       LDR     r5,[r0,#0x10]
1       STR     r4,[r5,#0x04]
```

```
15 cycles for HW Int. Response
5 cycles to push 4 registers
11 cycles for instrs at 0x081e++
(from ARM CM0+ TRM Table 3-1)
Total = 31 cycles = 646 ns
```

- Analyze object code (covered in ECE 461/561)
  - Interrupt handler needs to run for 16 cycles before debug output signal is raised
  - CPU instruction pipelining delay? (2 stages = 2 cycles)
  - Output port delay (1 cycle?)

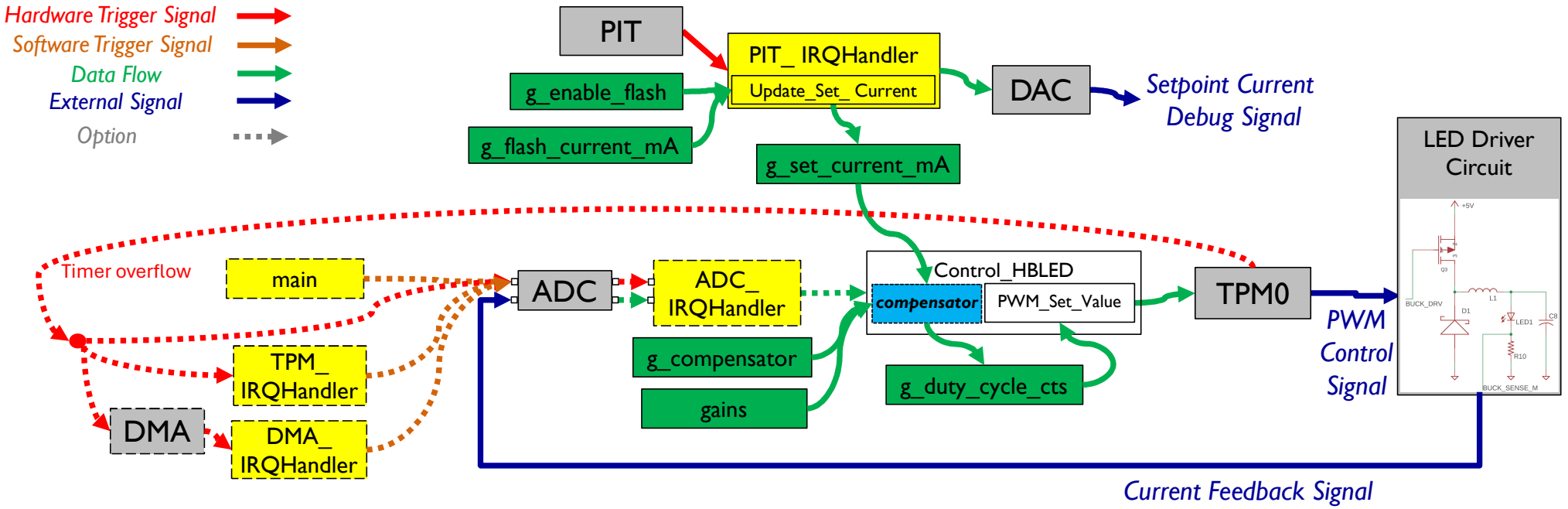
Impact of Changing Switching and Control Loop Frequencies

- As SMPS switching frequency f_{Sw} rises:
  - Size, weight, cost of inductors & capacitors falls
  - Switching power losses rise
- As control loop frequency f_{Ctl} rises:
  - Control quality rises
  - CPU time left for other processing falls
  - CPU MHz requirement rises, so cost, power also rise
- Useful to be able set f_{Sw} and f_{Ctl} to different values to customize system better.
  - f_{Sw} >> f_{Ctl} due since switching is faster and simpler than read inputs/calculate/update output
  - For synchronous sampling, set $f_{Ctl} = \frac{f_{Sw}}{N}$ with integer N
- Controller architecture allows direct control of f_{Sw} and integer N to get f_{Ctl}
  - Several implementations with different performance, costs and limits



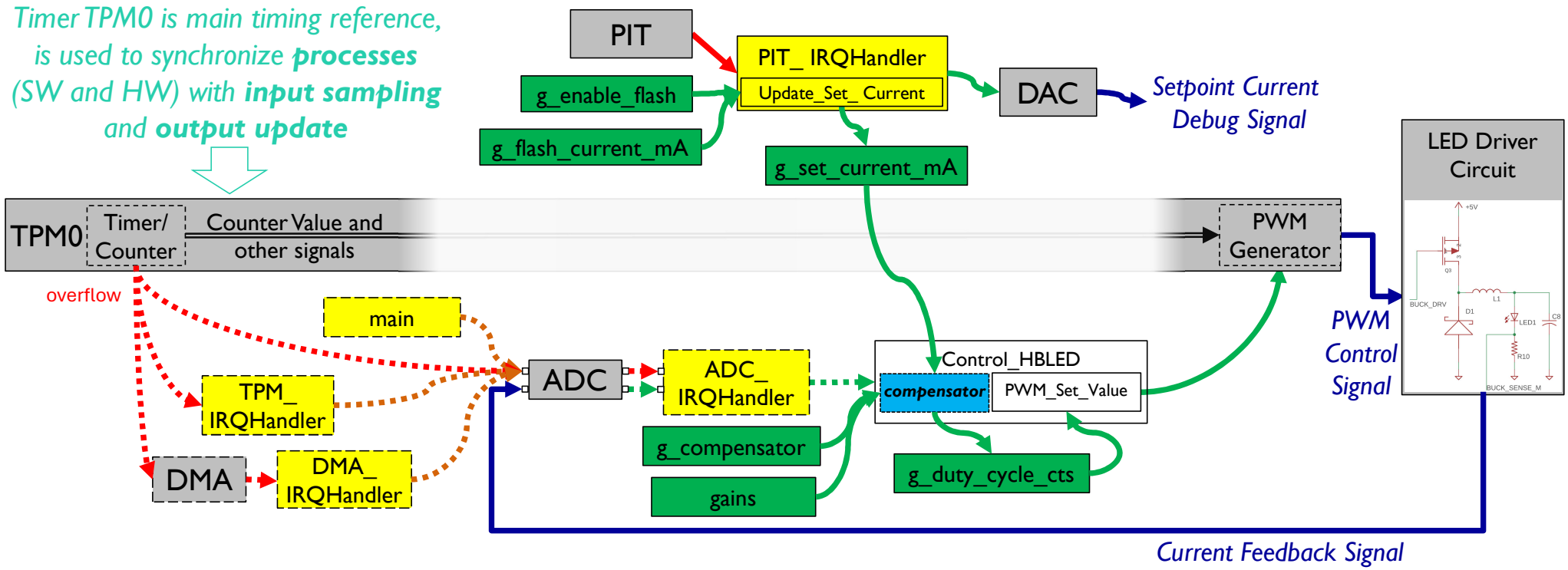
NC STATE UNIVERSITY

General Controller Architecture – View 1

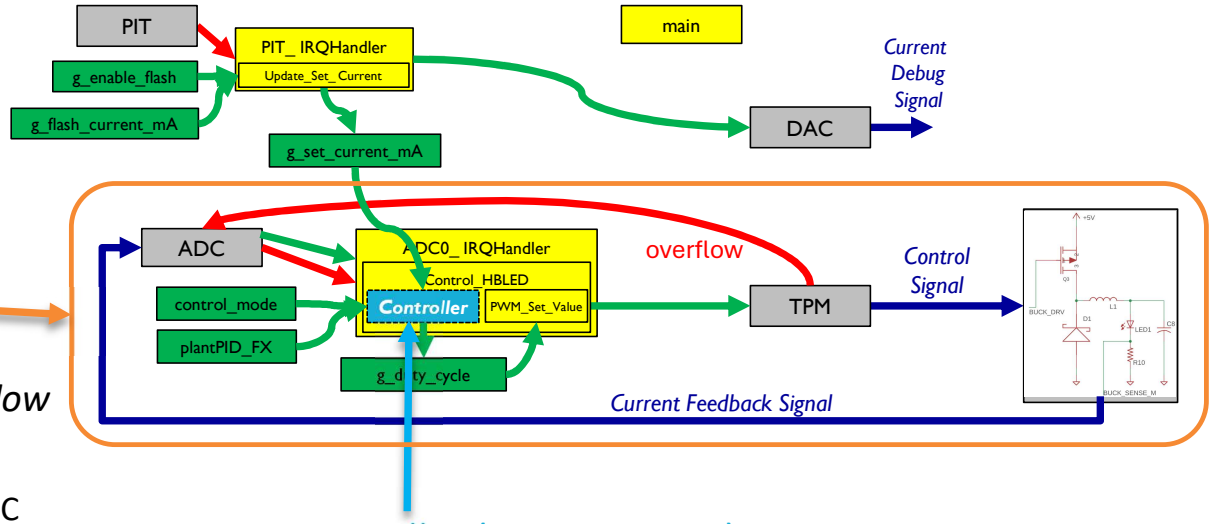


NC STATE UNIVERSITY

General Controller Architecture – View 2



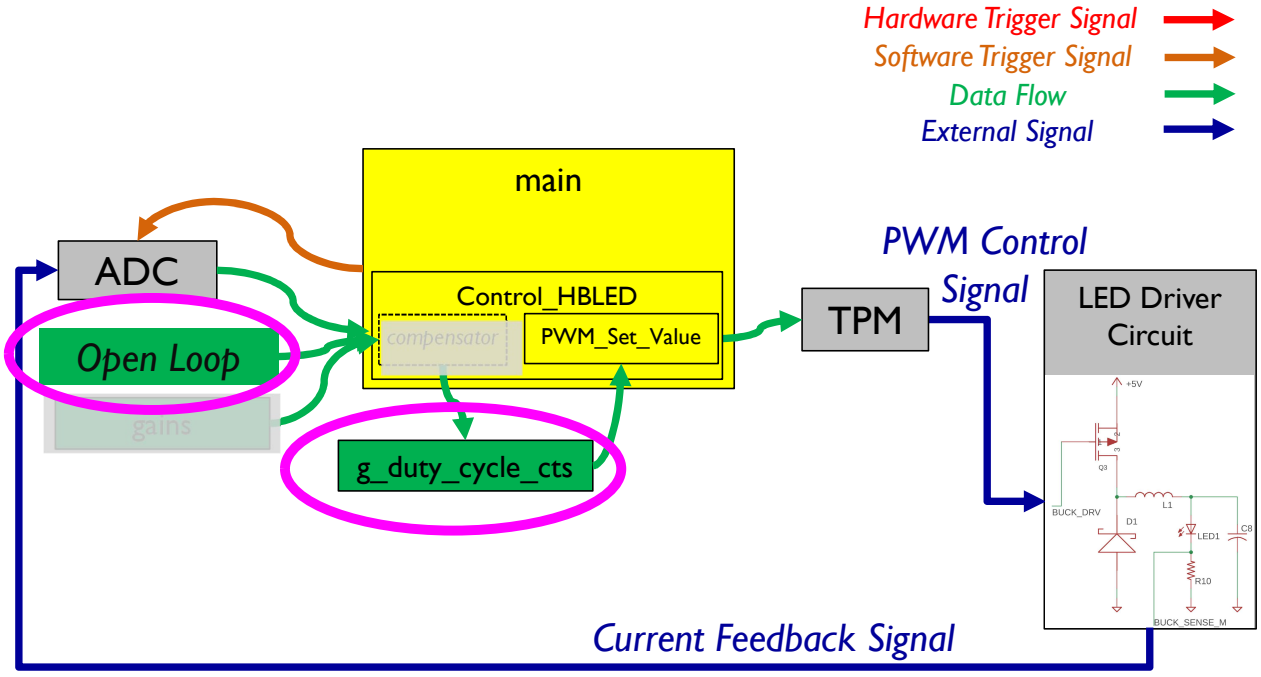
Configuration in HBLED.h

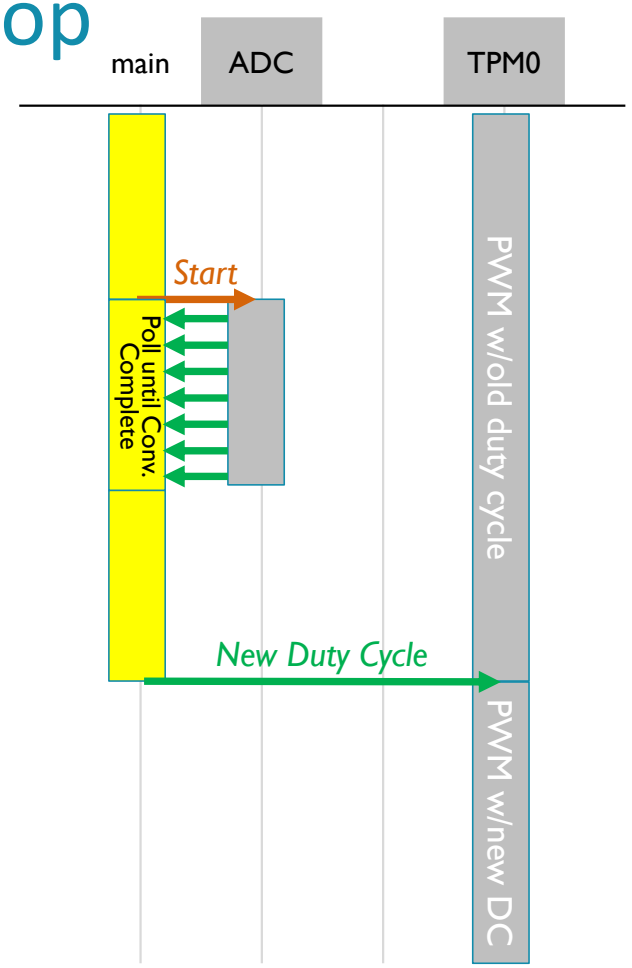


- Control architecture options
  - *"Hard-Coded": Rebuild code to change, since these run-time decisions are too slow*
  - Asynchronous sampling
    - Free-running S/W loop in main triggers ADC
  - Synchronous sampling
    - No **frequency division**: $f_{ctl} = f_{sw} = f_{TPM0}$
      - ADC hardware triggered by TPM0 overflow
    - Software **frequency division**: $f_{ctl} = f_{sw}/N = f_{TPM0}/N$
      - N set by SW_CTL_FREQ_DIV_FACTOR
      - ADC software triggered by TPM0 ISR
    - Dual timers: $f_{ctl} = f_{TPM2}$, $f_{sw} = f_{TPM0}$
      - TPM2 ISR enables TPM0 IRQ once
      - TPM0 ISR software triggers ADC, disables TPM0 IRQ

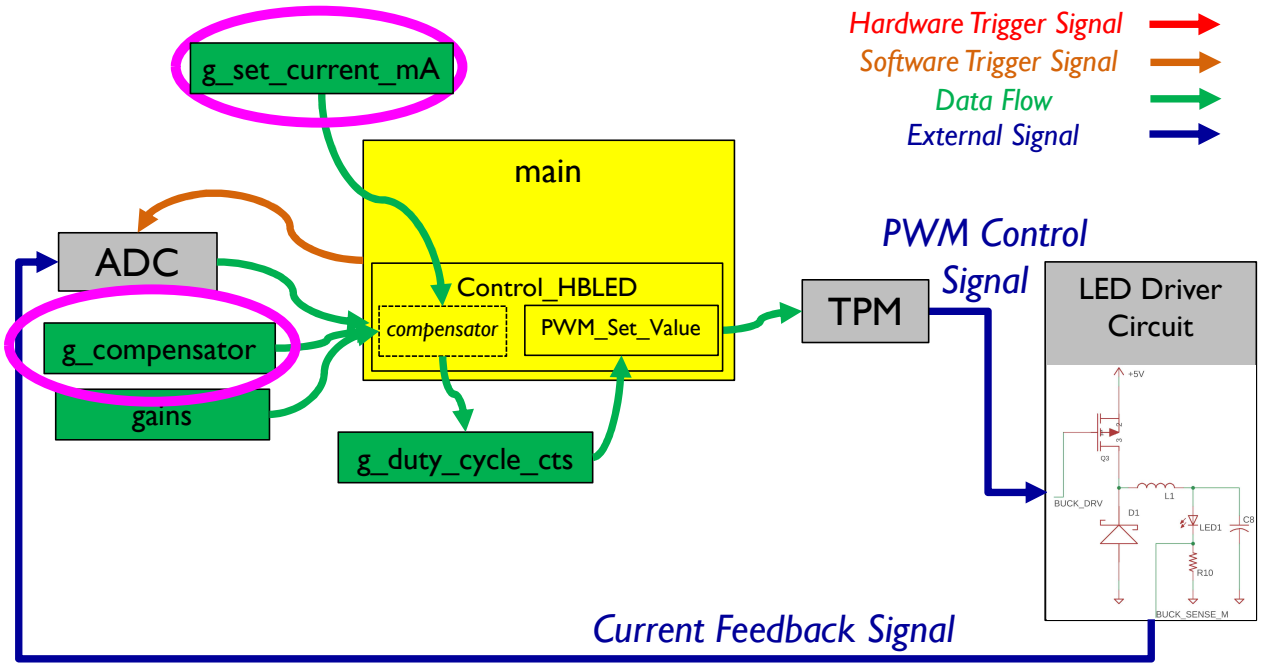
- Controller (compensator) options
  - *"Software-Configurable": Can change compensator and gains by tweaking variables as program runs*
  - Types
    - None (open loop)
    - Bang-Bang, Incremental, PID, Fixed-point PID
  - Parameters
    - Gains are scaled by control loop period (faster update rate reduces the required gain)

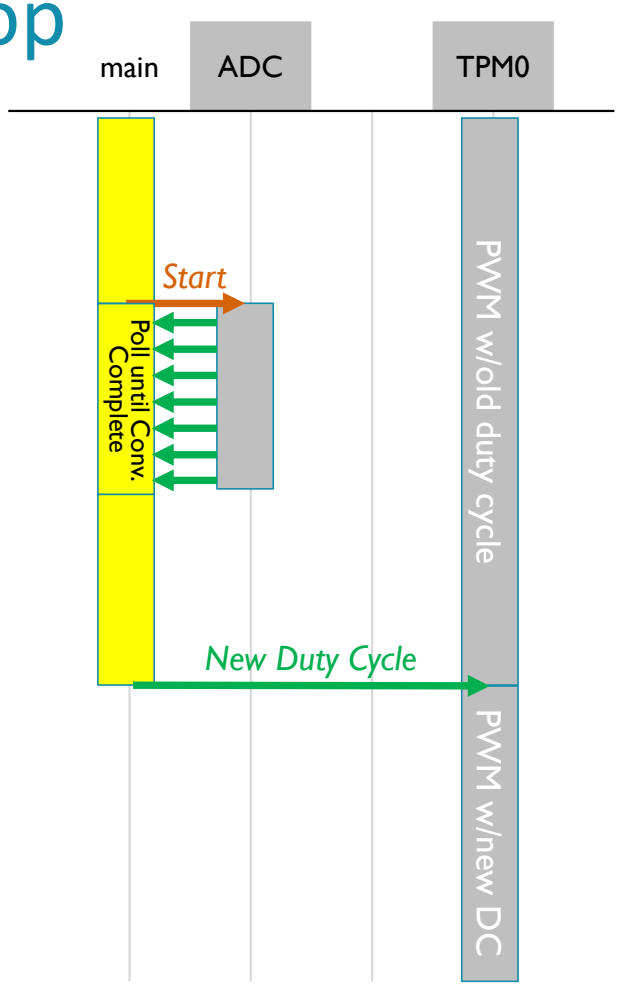
Q1-6: Asynchronous Sampling, Open Loop



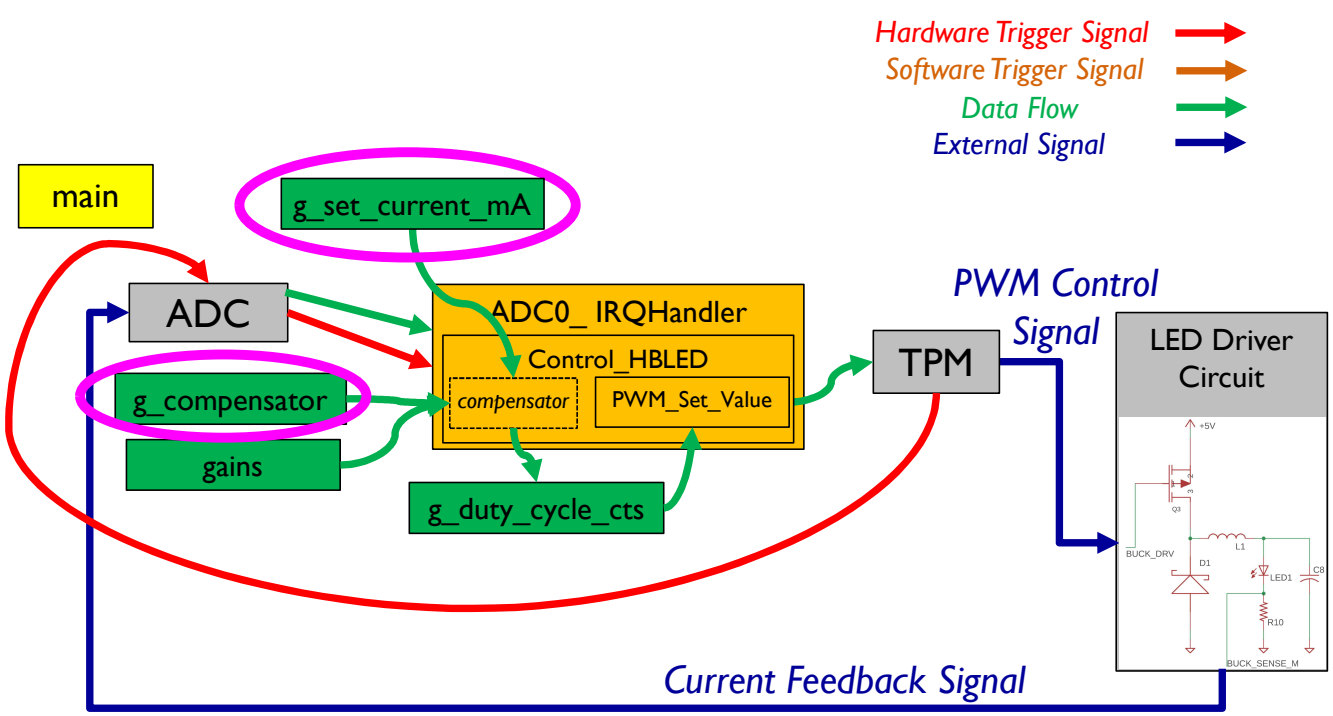


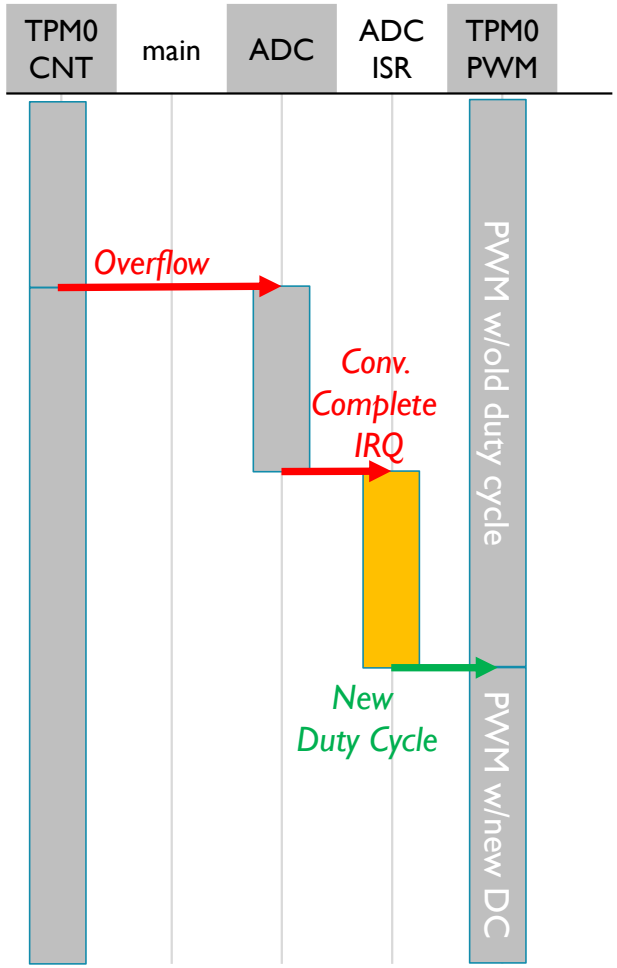
Q7: Asynchronous Sampling, Closed Loop





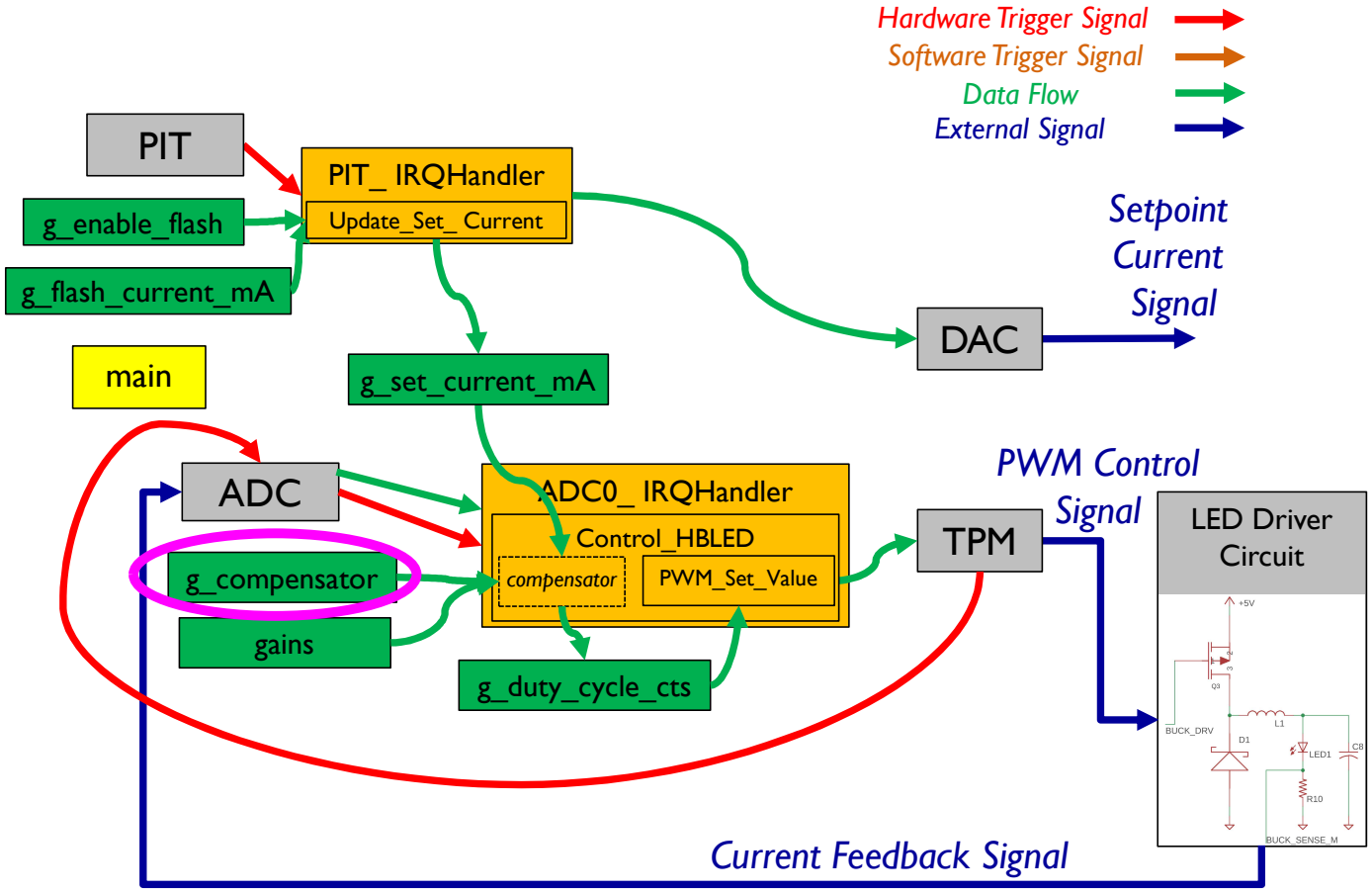
Q8-12: Synch. Sampling, Closed Loop

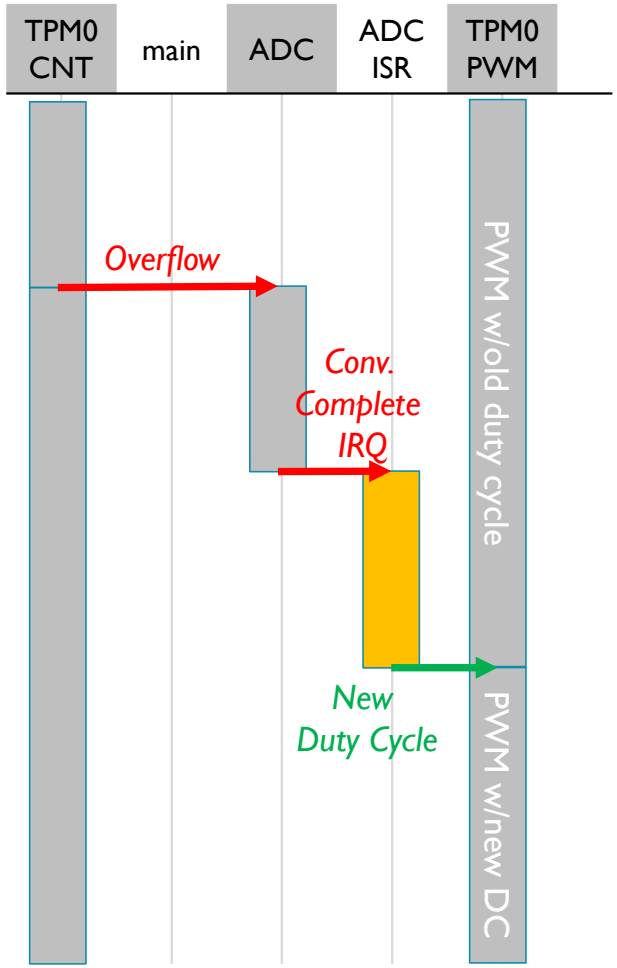




Q13-14: Synch., Closed Loop, Transients

NC STATE UNIVERSITY



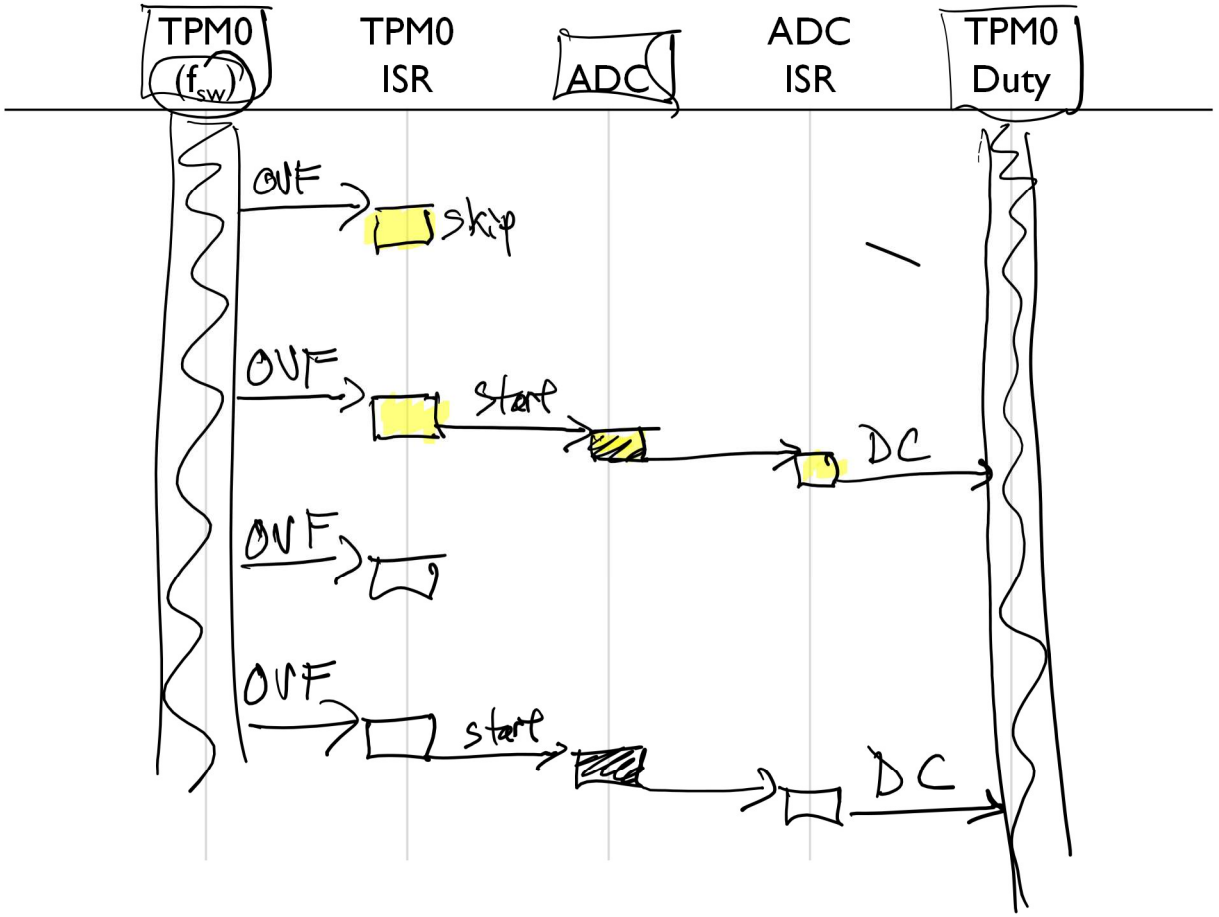


NC STATE UNIVERSITY

SYNC Sequence Diagram, Software Division for F_{ctl}

USE_SYNC_SW_CTL_FREQ_DIV

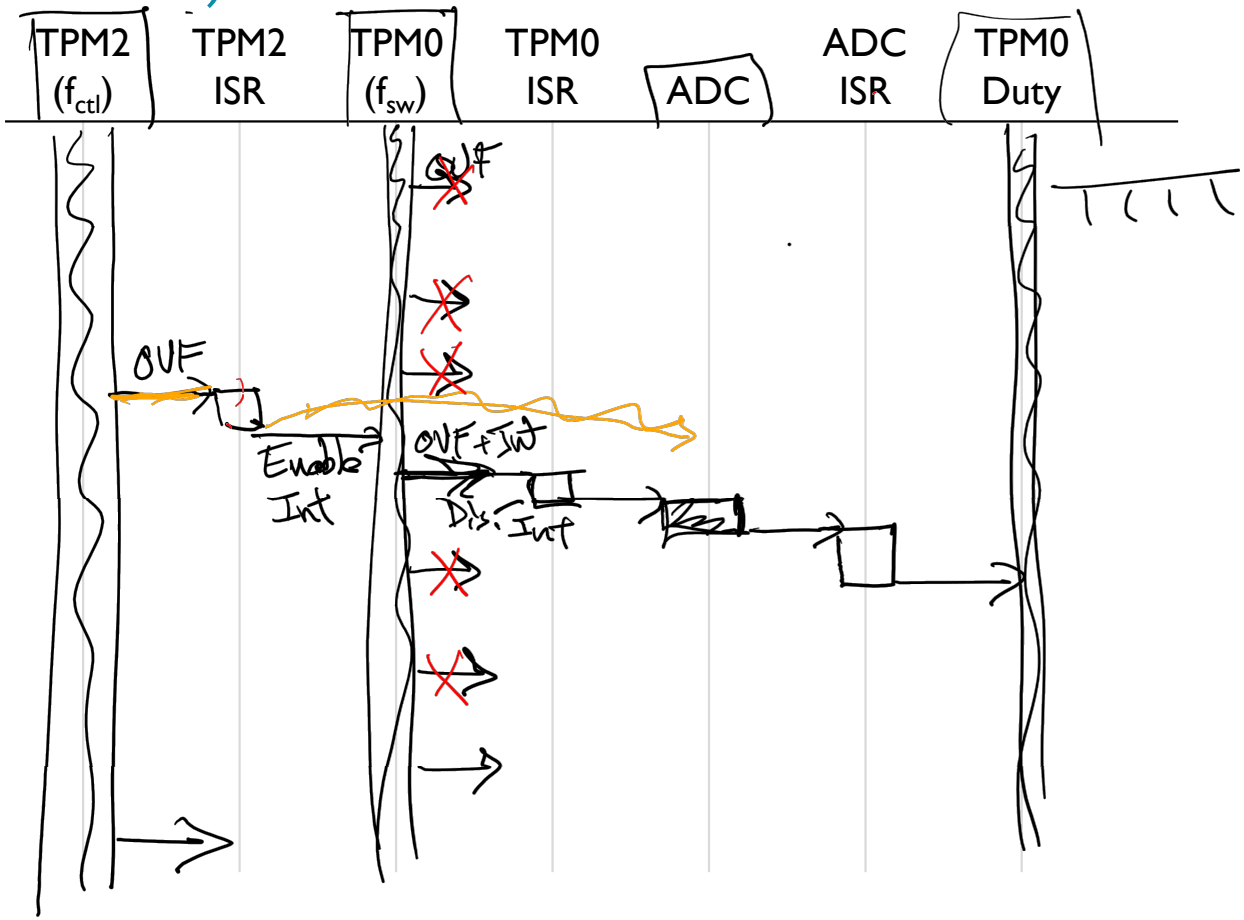
$f_{sw} = 2 \cdot f_{ctl}$



NC STATE UNIVERSITY

SYNC Sequence Diagram, 2nd Timer for F_{ctl}, ADC SW Trigger

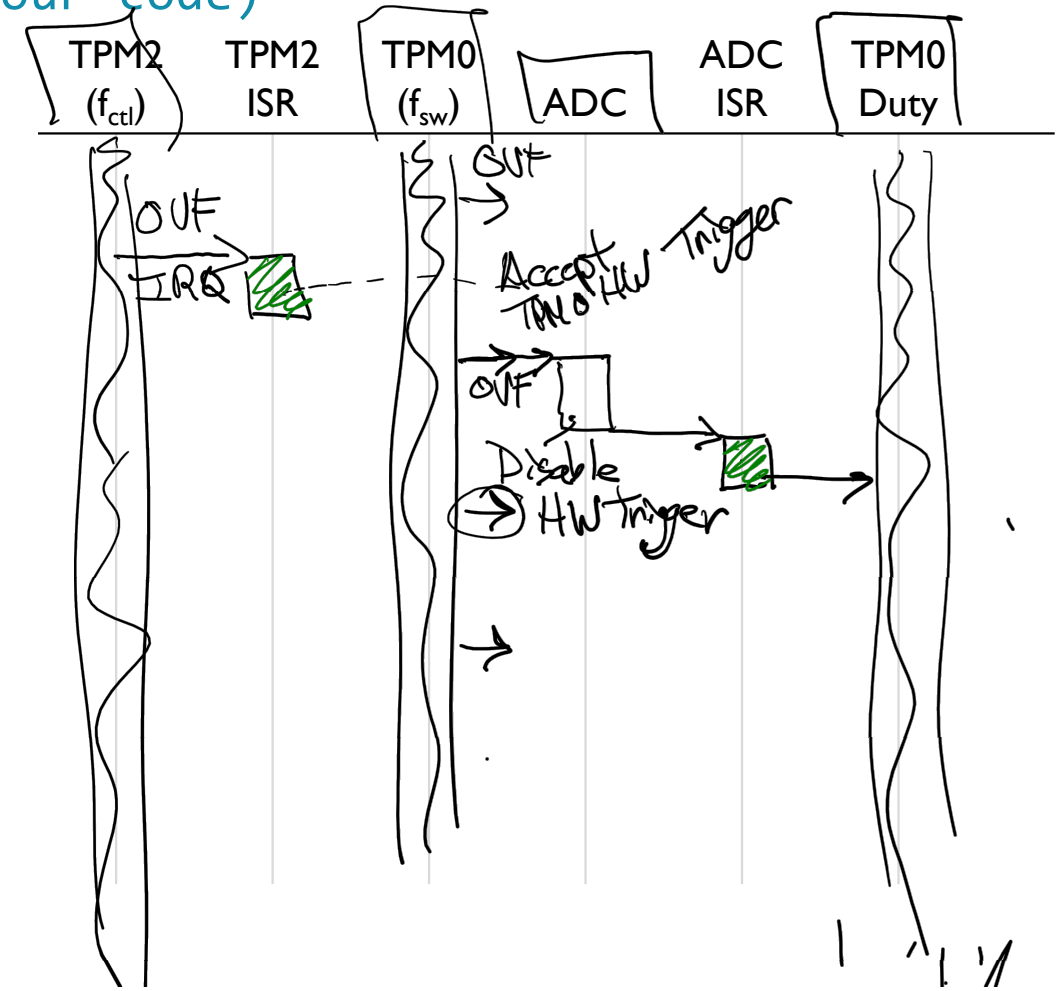
(not in your code)



NC STATE UNIVERSITY

SYNC Sequence Diagram, 2nd Timer for F_{ctl}, ADC HW Trigger

(not in your code)



How About Using DMA?

(not in your code)

- Replace TPM2 ISR with DMA transfer to enable ADC hardware trigger