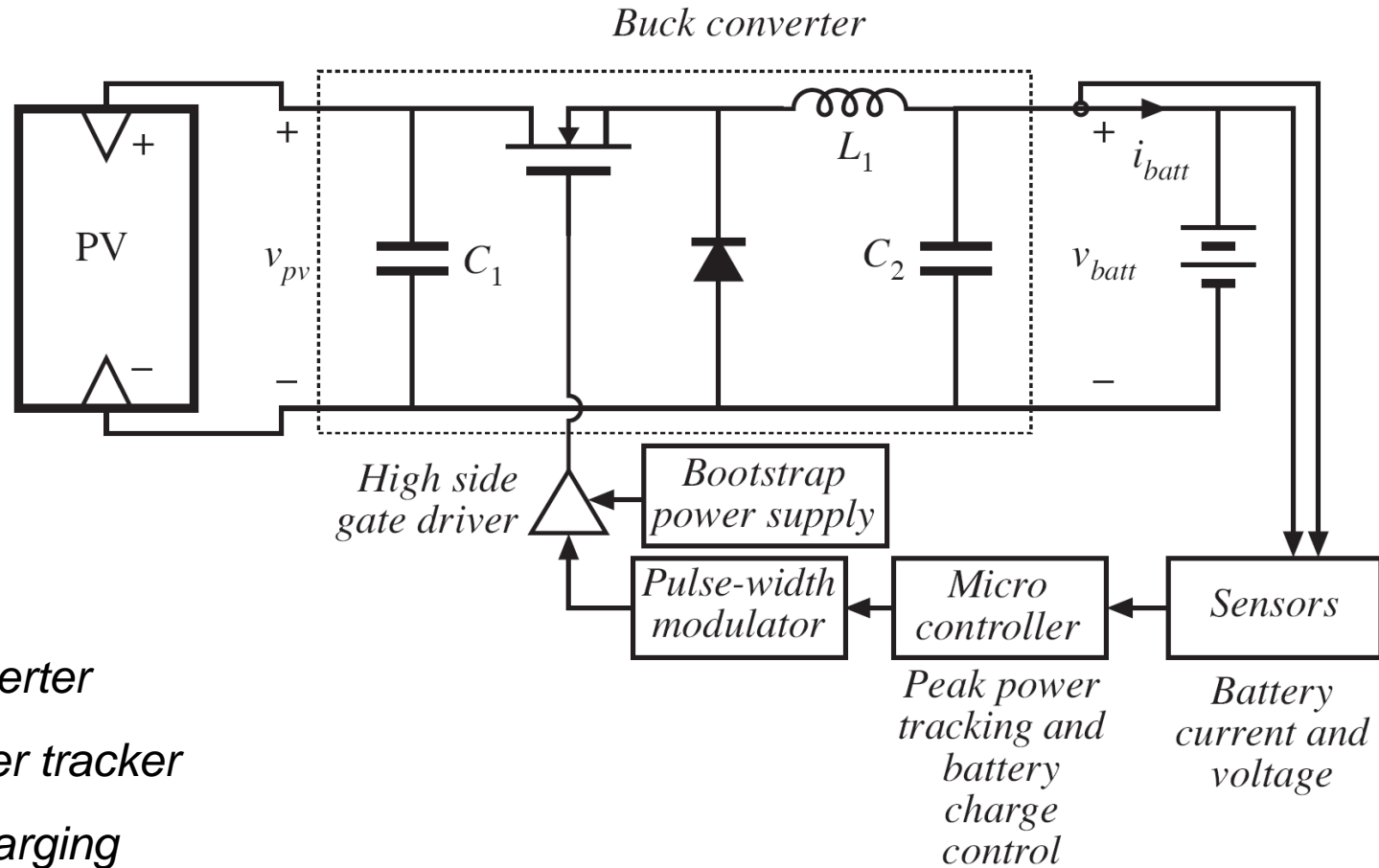


# Lecture 5

## ECEN 4517/5517



# Goals in upcoming weeks

---

## This week (Exp. 3, week 2):

Finish Exp.3 part 1 with buck converter operating outside with PV panel

Start Exp.3 part 2: digital control with the MSP430 microcontroller

## Next week (Exp. 3, week 3):

Demonstrate peak power tracker algorithms outside with converter connected between the PV panel and battery

## Following week (Exp. 4, week 1):

Finish Exp. 3 if not complete

Start inverter experiment

# What's due this week & next

---

Right now:

Prelab assignment for Exp. 3 week 2 (one from every student):  
Battery current and voltage sensing circuit details

Next week in lecture:

Prelab assignment for Exp. 3, week 3 (one from every student):  
proposed (not experimentally validated) **void main ( void )**  
code for peak power tracking

Late assignments will not be accepted.

# Experiment 3

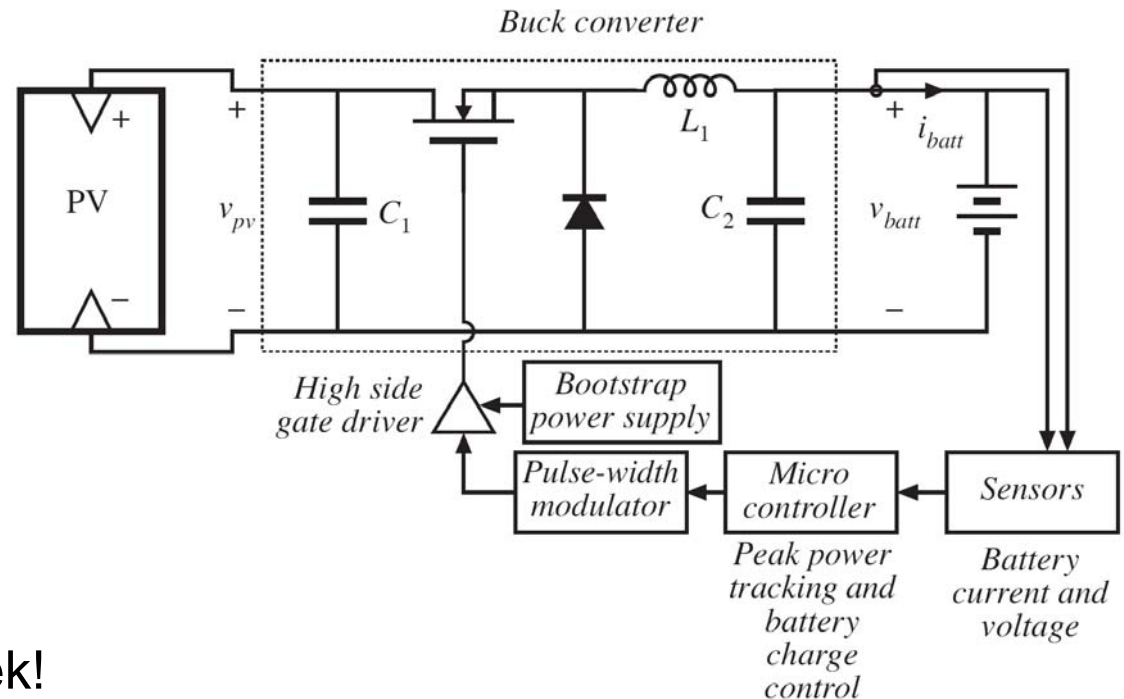
## Exp. 3 Part 1:

Demonstrate buck converter power stage operating open loop

Inside, with input power supply and resistive load

Outside, between PV panel and battery

DC system simulation



## Exp. 3 Part 2 – Start this week!

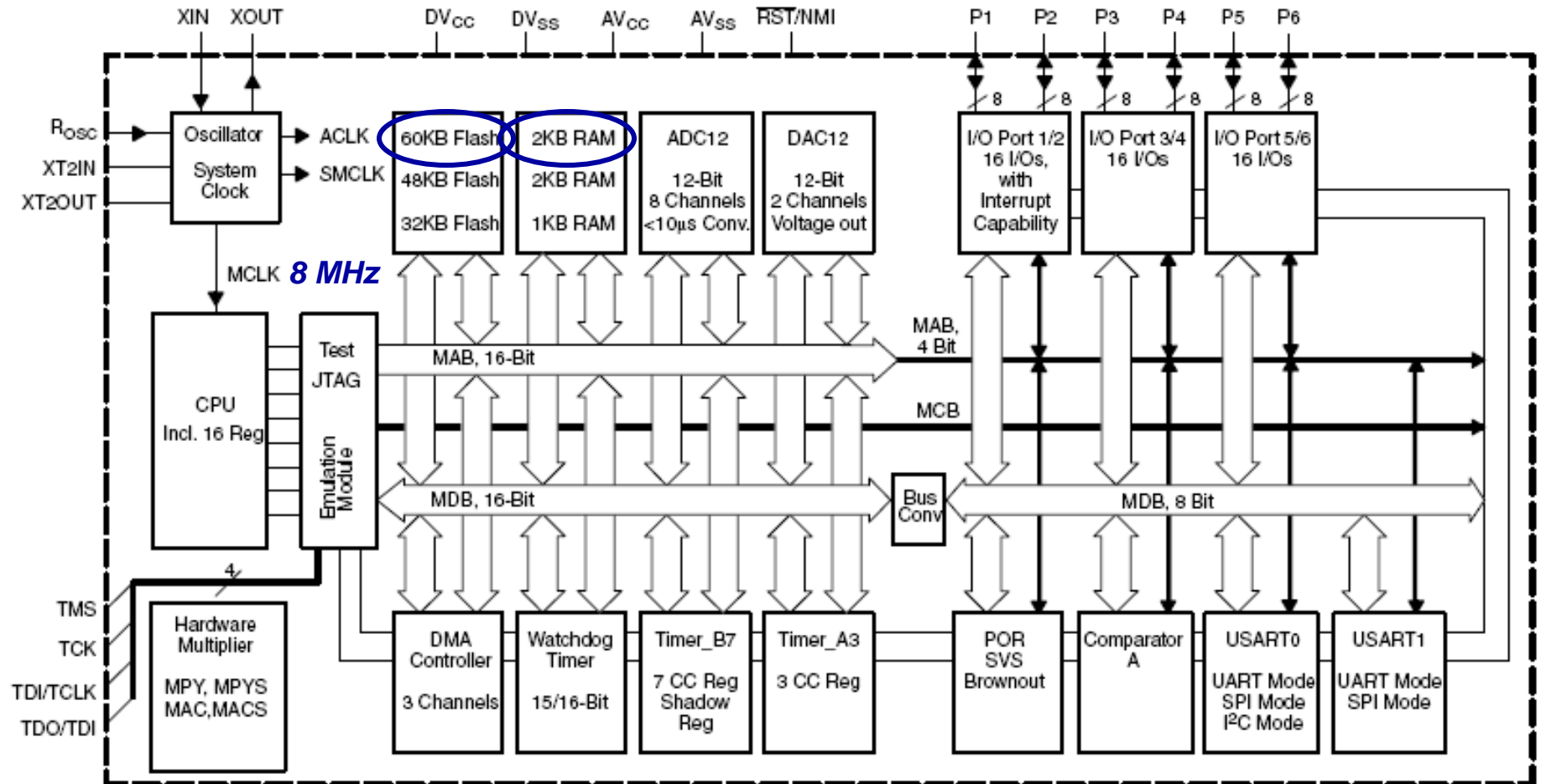
Demonstrate open-loop control of converter from microprocessor

Demonstrate working sensor circuitry, interfaced to microprocessor

Demonstrate peak power tracker algorithms outside with converter connected between the PV panel and battery

# TI MSP430 Microcontroller

## MSP430x16x



# Analog-to-Digital Converters

---

- There are eight (multiplexed) A/D channels, labeled A0 through A7 in the pinout diagram. Some of these pins are shared with other functions.
- A/D channels A6 and A7 will not be used because their pins are used for D/A outputs instead.
- A/D channels A0 and A1 will not be used because the development board hard-wires their pins to the LED and button instead.
- A/D channels A2 through A5 are available as usable A/D inputs.
- A/D channel A2 is connected to the connector pin labeled P6.2. In the project files, the sampled voltage appears in the variable *Adc\_data*[0].
- A/D channels A3, A4, and A5 are connected to connector pins P6.3, P6.4, and P6.5 respectively. In the project files, the sampled voltages of these pins appear in the variables *Adc\_Data*[1], *Adc\_Data*[2], and *Adc\_Data*[3].
- These inputs convert voltages in the range 0 to 2.5 V. Do not apply voltages exceeding 3.3 V or negative voltages to these pins!

# Additional Peripherals

---

## Digital-to-analog converter

- There are two D/A converter channels: DAC0 and DAC1.
- Channel DAC0 is connected to pin P6.6 of the connector. In the project files, the voltage at this pin is controlled by writing to the DAC0 data register *DAC12\_0DAT*.
- Channel DAC1 is connected to pin P6.7 of the connector. In the project files, the voltage at this pin is controlled by writing to the DAC1 data register *DAC12\_1DAT*.

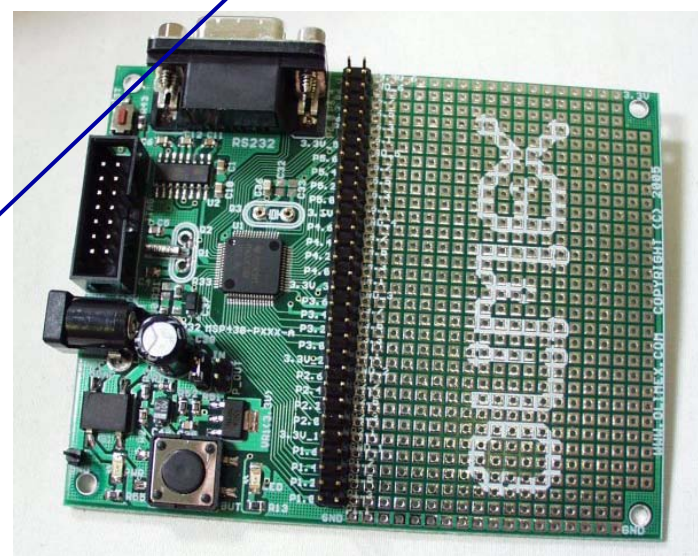
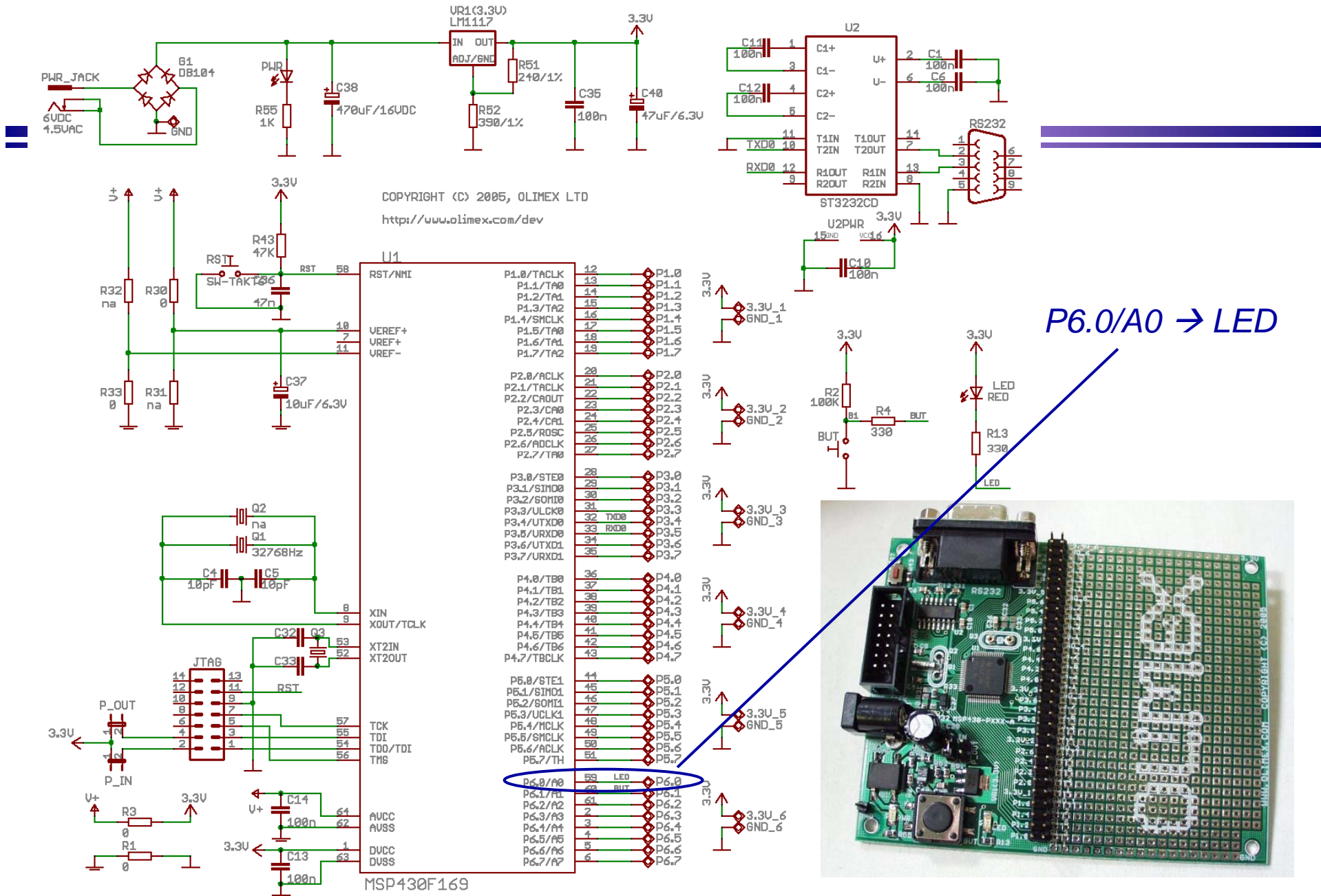
## Timer / PWM

- Timer A is used to generate pulse-width modulated waveforms.
- In the project files, Timer A output TA1 is used to output a PWM waveform which appears on pin P1.6. The duty cycle of this waveform is controlled by writing to the global variable *D0*.
- In the project files, Timer A output TA2 is used to output a PWM waveform which appears on pin P1.7. The duty cycle of this waveform is controlled by writing to the global variable *D1*.

## LED and button

- There is a small LED on the development board, which is connected to pin P6.0. The LED can be controlled by writing to bit 0 of peripheral 6.
- The pushbutton on the development board is connected to pin P6.1. The state of the pushbutton can be read in a similar manner.

# Olimex Dev Board Schematic



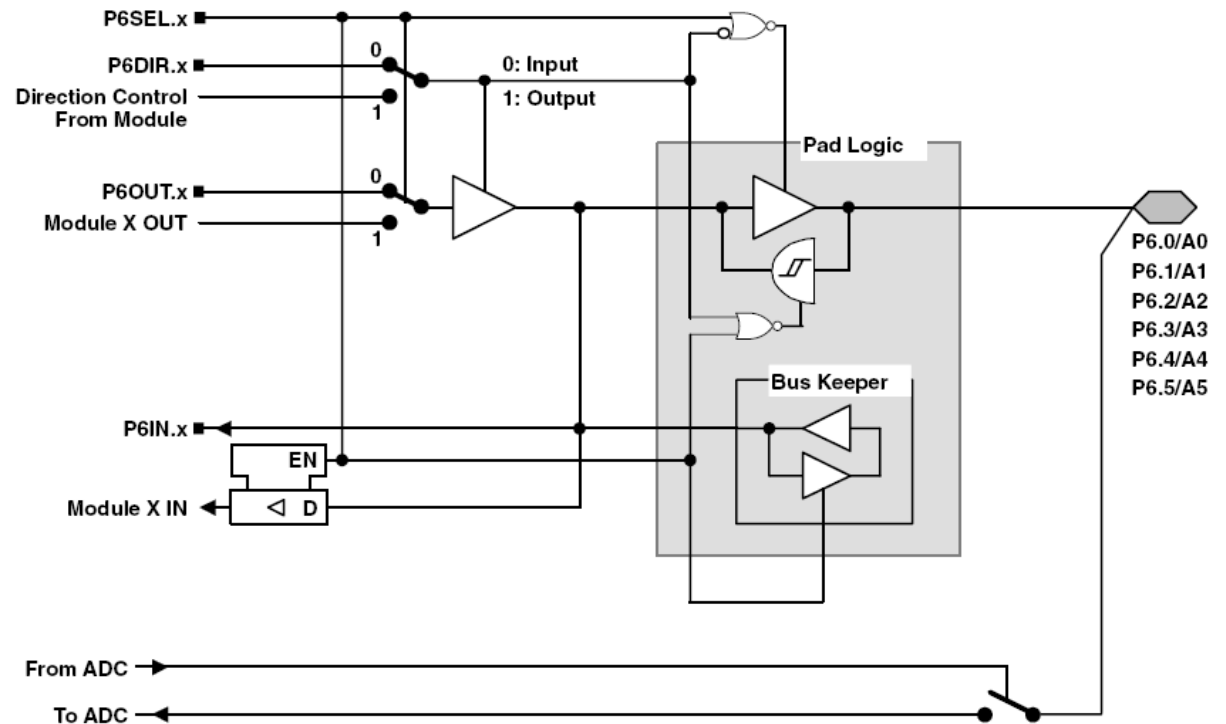
# Project Software Files

---

- main.c***: contains the system startup routine and the main system loop. This is where all of your code goes. You can access all 4 active ADC channels, `Adc_Data[x]`, `DAC0`, `DAC1`, `D0`, `D1`, the LED (on `P6OUT`, bit 0) and the push-button.
- periph.c***: contains configuration subroutines for all used peripheral devices as well as some routines for starting and stopping peripherals.
- periph.h***: provides function prototypes for the externally callable functions in `periph.c`.
- config.h***: contains system and board constants and some typedefs.
- msp430x16x.h***: is TI's provided include file and contains `#define`'d register names and constants for the MSP430. All of the names match those in the family datasheet.

# Configuring the I/O Ports

port P6, P6.0 to P6.5, input/output with Schmitt-trigger



- Setup in systemInit, main.c
- Configure input/output, e.g.

// Port6:

// P6.0: LED output

// P6.2-6.5: ADC2-ADC5 inputs

// P6.6-6.7: DAC0 and DAC1 outputs

P6SEL = BIT2 + BIT3 + BIT4 + BIT5 + BIT6 + BIT7;

P6DIR = BIT0 + BIT6 + BIT7;

# Tutorials

---

## **Tutorial 1: LED**

- Getting started with the programming environment (IAR)
- Hardware setup, Make/Run code, debugging steps and breakpoints, blink an LED

## **Tutorial 2: PWM controller**

- Modify the project software files to realize a digital PWM
- Utilize ADC, DAC and Timer/PWM peripherals

# Exp. 3, Part 2: Digital PWM

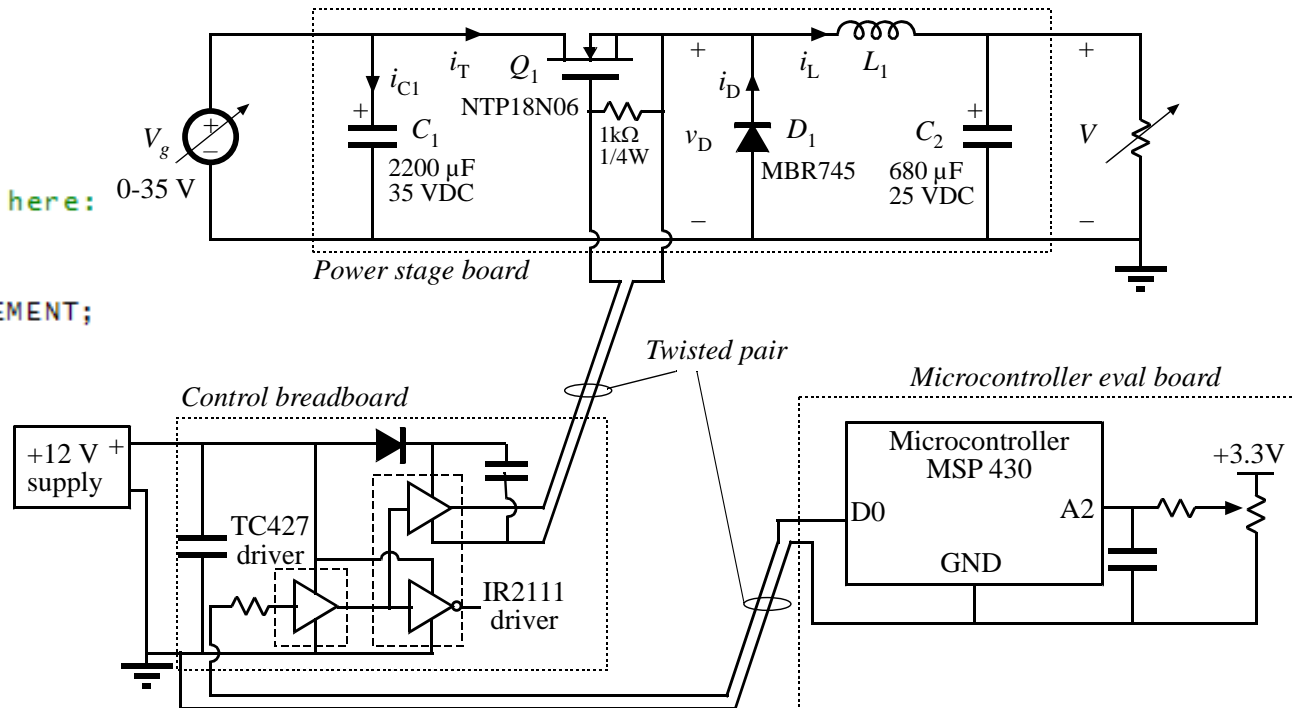
```
#define PWM_INCREMENT (4096/80)
```

```
void main( void )
{
    uint16 temp_adc0;
    uint16 d;

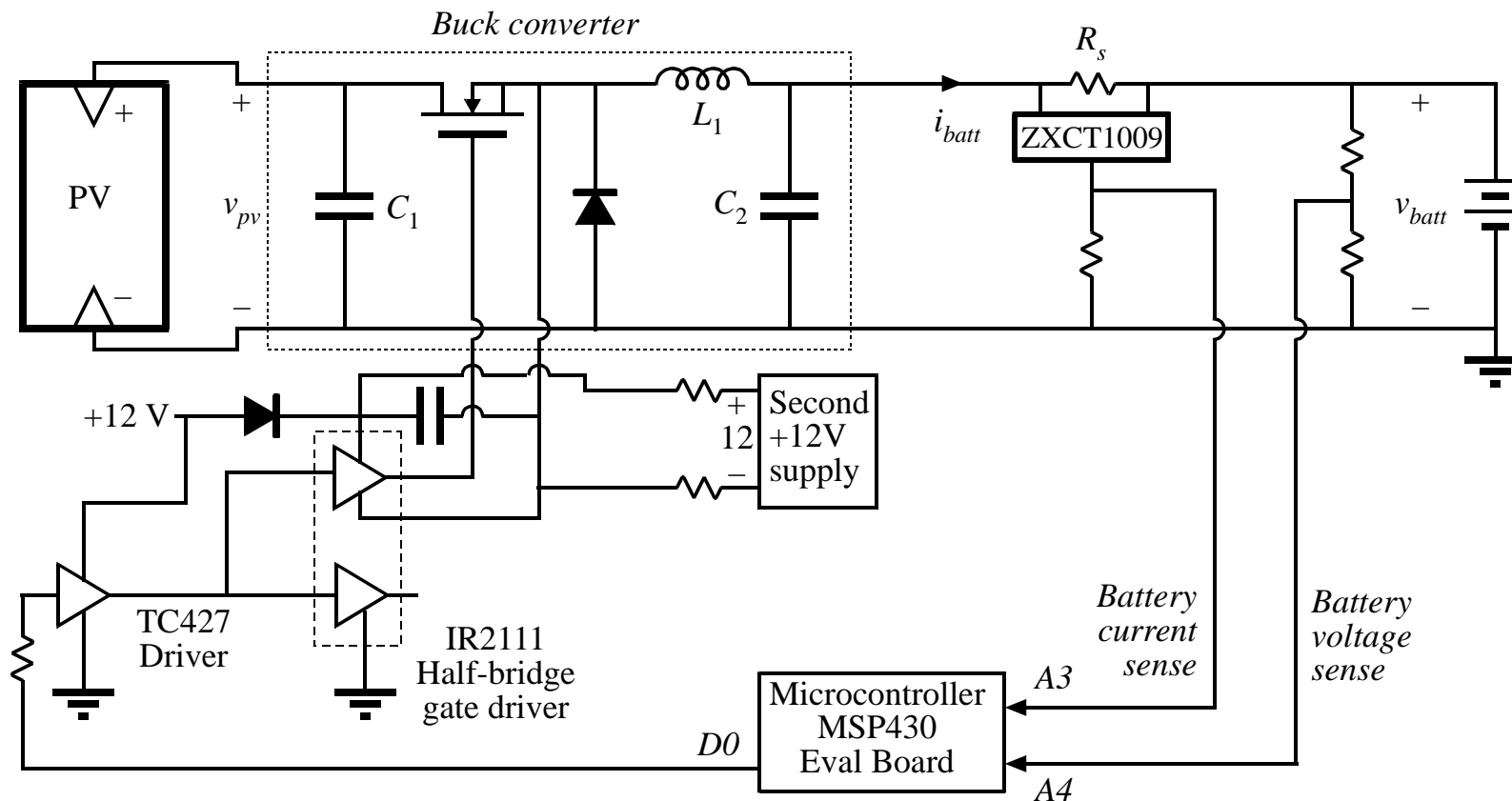
    systemInit();

    while (1)
    {
        // Control algorithms go here:
        temp_adc0 = Adc_Data[0];
        d = temp_adc0 / PWM_INCREMENT;

        if(d < 8)
        {
            d = 8;
        }
        else if(d > 72)
        {
            d = 72;
        }
        D0 = d;
    }
}
```



# Exp. 3 Part 2: PV Peak Power Tracking Current & Voltage sensing



*Be careful with twisted pair lines, grounding, filtering, MSP430 protection, etc.*

# Maximum Power Point Tracking: Perturb and Observe (P&O)

- A well-known approach
- Works well if properly tuned
- When not well tuned, maximum power point tracker (MPPT) is slow and can get confused by rapid changes in operating point
- A common choice: “control” is switch duty cycle

