ECEN 4517/5517 Lecture 2

- Upcoming assignments due:
  - Exp. 1 final report due in D2L dropbox by 6:00 pm Friday Jan. 31
  - Exp. 3 part 1 prelab assignment due in D2L dropbox by Monday, Feb. 3 at 1pm

- This week: Exp. 2: Introduction to MSP 430
  - Lab kits are available in ECEE Electronics Store, ECEE 1B10
    - You will need this kit to perform Exp. 2.
    - You will also need two oscilloscope voltage probes, as well as small parts (capacitors) from the undergraduate electronics lab parts kit.

- Next week in lecture [note change of date]
  - 10 minute quiz on Exp. 1. See Exp. 1 web page for a sample
Lab reports

- One report per group. Include names of every group member on first page of report.
- Report all data from every step of procedure and calculations. Adequately document each step.
- Discuss every step of procedure and calculations
  - Interpret the data
  - It is your job to convince the grader that you understand what is going on with every step
  - Regurgitating the data, with no discussion or interpretation, will not yield very many points
  - Concise is good
  - Annotate diagrams and plots, use consistent notation
Upcoming weeks:
Design and build MPPT system

Exp. 3: DC-DC converter

Exp. 2: introduction to MSP430 microcontroller

Measure voltage, current
This Week’s Experiment 2

• Become familiar with MSP430
• Set up your MSP 430 to drive a MOSFET at a programmable duty cycle
• Discussion: online Quizzes 2.1 and 2.2
Introduction to MSP430
ECEN 4517/5517

• MSP430 microcontroller
• MSP430 development board
• Simple code examples
  • Timer D, high-resolution digital pulse-width modulation
  • Analog-to-Digital conversion (ADC)
Experiments 2 and 3
Design and build MPPT system

Exp. 2: introduction to MSP430 microcontroller

Exp. 3: DC-DC converter

DC-DC

Gate driver

Measure voltage, current

Microcontroller
MSP430F5172: Resources

MSP430F5172 User’s Guide
  • The primary resource for operation and programming of on-chip peripherals (PWM, ADC, etc.)
  • Linked to Exp. 2 web page, 1147 page pdf

MSP430F5172 Data Sheet
  • Describes pinouts, specifications
  • Linked to Exp. 2 web page, 103 page pdf

Code Composer Studio 5.3
  • Development system for MSP430; program in C
  • On lab computers: free version (limited code size)

Library of Code Examples
  • Accessible within Code Composer Studio, also linked to web page
  • Many programming examples for each peripheral
  • Use directory of examples for 430F5172 chip
  • Also: Erickson’s sample code main.c linked to Exp. 2 web page
Experiment 2
Introduction to MSP 430F5172 Microcontroller

Clocks: ACLK, SMCLK, MCLK
Up to 25 MHz @3.3V
CPU: 16 bit

Programmable multi-use I/O ports (31)

ECEN 4517
Microcontroller Pinout

ADC inputs

A0 to A5, A7, A8

to LED (P1.0 output)
P.x.y is digital I/O

P1.0/PM_UCA0CLK/PM_UCB0STE/A0*/CB0
P1.1/PM_UCA0TXD/PM_UCA0SIMO/A1*/CB1
P1.2/PM_UCA0RXD/PM_UCA0SOMI/A2*/CB2
P1.3/PM_UCB0CLK/PM_UCA0STE/A3*/CB3
P1.4/PM_UCB0SIMO/PM_UCB0SDA/A4*/CB4
P1.5/PM_UCB0SOMI/PM_UCB0SCL/A5*/CB5

DA PACKAGE (TOP VIEW)

PM_TD0.x
PM_TD1.x
See also pins 20, 23-28, 33
Microcontroller default settings

Upon power-on reset (POR), the MSP430F5172 comes up with the following conditions:

- Watchdog timer is enabled
- All pins are set to read state
- Processor internal clock and core voltage are set to minimum values. Default clock frequency = 1.045 MHz

- Processor supply voltage is 3.3 V
- Internal processor core operates at lower voltage; a programmable internal voltage regulator reduces the 3.3 V to this lower voltage
- Faster clock speeds require higher core voltages
- Digital I/O pins can operate at 5 V if 5 V is supplied to DVIO pin. Otherwise, these pins operate with 3.3 V logic levels. In the lab, we will always work with 3.3 V supply and 3.3 V logic levels.
Development board
in your kit

- **External power**
- **JTAG (to computer)**
- **Jumper:**
  - Select power source—JTAG (as shown) or external
- **Header:**
  - Processor pins 1-19
- **Header:**
  - Processor pins 20-38
- **Jumper:**
  - Connect or disconnect LED from P1.0
- **Jumper:**
  - Select digital I/O power—Internal 3.3 V (place jumper to short 1-2, as highlighted), or external 5 V
- **Reset button**
Development board schematic: header pinout
(complete board schematic linked on the Exp 2 page)
Peripherals are controlled by registers in addressable memory

**Example:** Port P1, comprised of eight pins labeled P1.0 – P1.7. Digital input/output

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Register Name</th>
<th>Type</th>
<th>Access</th>
<th>Reset</th>
<th>Section</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1IN or PAIN_L</td>
<td>Port 1 Input</td>
<td>Read only</td>
<td>Byte</td>
<td></td>
<td>Section 12.4.9</td>
<td>Read input value</td>
</tr>
<tr>
<td>P1OUT or PAOUT_L</td>
<td>Port 1 Output</td>
<td>Read/write</td>
<td>Byte</td>
<td>undefined</td>
<td>Section 12.4.10</td>
<td>Write output value</td>
</tr>
<tr>
<td>P1DIR or PADIR_L</td>
<td>Port 1 Direction</td>
<td>Read/write</td>
<td>Byte</td>
<td>00h</td>
<td>Section 12.4.11</td>
<td>0 = input 1 = output</td>
</tr>
<tr>
<td>P1REN or PAREN_L</td>
<td>Port 1 Resistor Enable</td>
<td>Read/write</td>
<td>Byte</td>
<td>00h</td>
<td>Section 12.4.12</td>
<td>When input, 1 = pullup/down</td>
</tr>
<tr>
<td>P1DS or PADS_L</td>
<td>Port 1 Drive Strength</td>
<td>Read/write</td>
<td>Byte</td>
<td>00h</td>
<td>Section 12.4.13</td>
<td>0 = reduced 1 = full drive</td>
</tr>
<tr>
<td>P1SEL or PASEL_L</td>
<td>Port 1 Port Select</td>
<td>Read/write</td>
<td>Byte</td>
<td>00h</td>
<td>Section 12.4.14</td>
<td>0 = GPIO 1 = selected for peripheral module</td>
</tr>
</tbody>
</table>

For further documentation, see *MSP430x5xx/6xx Family User Guide*, Ch 12, pp. 406ff.

There are additional P1 registers related to interrupts.

TI provides a header file that sets up all registers with C code variable names assigned to the correct addresses. Just add the following statement to the beginning of your C code:

```
#include <msp430f5172.h>
```

This file also defines constants that are useful for setting peripheral functions.
Simple Code Examples

Configure pin P1.0 to be a digital output, and toggle its value

P1DIR |= 0x01; // OR the contents of register P1DIR with hex 01,
// forcing the first bit high
// This configures pin P1.0 to be an output

P1OUT ^= 0x01; // EXOR the contents of P1OUT with hex 01,
// toggling the first bit
// This changes the state of logic output P1.0

Turn off the watchdog timer

WDTCTL = WDTPW + WDTHOLD; // Sets the WDT control register WDTCTL to
// disable the watchdog timer function
// WDTPW and WDTHOLD are constants defined
// in the header file supplied by TI—see user guide
C code to toggle pin P2.2

Code example: drive P2.2 (pin 19) with a low-frequency square wave. The development boards have an LED connected to P1.0; if the code is modified to drive P1.0 then it will blink the LED. This is your Task 1 in Exp 1.

```c
#include <msp430.h>
/
* main.c
* Drive pin P2.2 with a low-frequency square wave
*/

void main(void) {
    volatile unsigned int i;  // Declare counter variable
    WDTCTL = WDTPW | WDTHOLD;  // Stop watchdog timer
    P2DIR |= 0x04;  // Configure pin P2.2 to output direction
    for (;;) {
        // Infinite loop
        P2OUT ^= 0x04;  // Toggle P2.2 output
        i = 10000;
        do(i--);
        while(i != 0);  // Wait 10000 cycles
    }
}
```
Introduction to MSP430
ECEN 4517/5517

• MSP430 microcontroller
• MSP430 development board
• Simple code examples
• Timer D, high-resolution digital pulse-width modulation
• Analog-to-Digital conversion (ADC)
Digital pulse-width modulation using MSP430 Timer D

Switching period $T_s = 1/f_s = nT_{clk}$, duty cycle $D = k/n$

$f_{clk} = 200$ MHz = TDCLK frequency, $T_{clk} = 5$ ns
High-resolution Timer D PWM setup

1. Setup core voltage and core clock frequency (SMCLK) to the highest values:
   - Level 3 core voltage = 3.3 V
   - SMCLK frequency = 25 MHz

2. Setup high-resolution Timer D generator to multiply SMCLK by 8 to TDCLK frequency = 200 MHz (5 ns time resolution)

3. Configure Timer D modes, switching period
1. Setting the core voltage and processor clock frequency

- The processor contains a digitally controlled oscillator (DCO) whose frequency can be programmed.

- Although the MSP430F5172 is powered with a 3.3 V supply, the processor core operates at a reduced voltage that can be programmed.

- Lower core voltage means less power dissipation but processor clock frequency is limited.

- At power-on reset: minimum core voltage (level 0) and low clock frequency (1.045 MHz)

- To operate at faster DCO frequency, we must raise core voltage one level at a time, then raise clock frequency. After each step, wait for circuitry to stabilize.

The numbers within the fields denote the supported PMMCOREVx settings.
Sample Code (main.c linked on the Exp 2 page)
Core voltage = level 3, processor frequency = 25 MHz

// Increase Vcore setting to level3 to support fsystem=25MHz
// NOTE: Change core voltage one level at a time...
SetVcoreUp (0x01);
SetVcoreUp (0x02);
SetVcoreUp (0x03);

// Initialize DCO to 25MHz
__bis_SR_register(SCG0); // Disable the FLL control loop
UCSCTL0 = 0x0000; // Set lowest possible DCOx, MODx
UCSCTL1 = DCORSEL_6; // Select DCO range 4.6MHz-88MHz operation
UCSCTL2 = FLLD_1 + 763; // Set DCO Multiplier for 25MHz
// (N + 1) * FLLRef = Fdco
// (762 + 1) * 32768 = 25MHz
// Set FLL Div = fDCOCLK/2
__bic_SR_register(SCG0); // Enable the FLL control loop

// Worst-case settling time for the DCO when the DCO range bits have been
// changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5172
// User Guide:
// 32 x 32 x 25 MHz / 32,768 Hz = 782000 = MCLK cycles for DCO to settle
__delay_cycles(782000);
Operation of Timer D as a PWM

See MSP430x5xx/6xx Family User Guide, Chapter 19

The MSP430F5172 has two Timer D’s
Each Timer D includes:
• One timer block with 16 bit counter
• Three capture/compare registers (CCR0 – CCR2)
• High resolution mode with TDCLK frequency = N*(DCO frequency)

Use CCR0 to set switching frequency: \( f_s = (\text{TDCLK freq})/(\text{CCR0}) \)
Use CCR1 and CCR2 to set duty cycles of outputs: D1 = CCR1/CCR0 etc.

Need to configure Timer D, and write values to set \( f_s \) and duty cycle(s)
Example: Configuring Timer D0 as a PWM with 100 kHz switching frequency  
(main.c linked on the Exp 2 page)

// Configure TimerD in Hi-Res Regulated Mode
TD0CTL0 = TDSSEL_2;  // TDCLK=SMCLK=25MHz=Hi-Res input clk select
TD0CTL1 |= TDCLKM_1; // Select Hi-res local clock
TD0HCTL1 |= TDHCLKCR; // High-res clock input >15MHz
TD0HCTL0 = TDHM_0 +  // Hi-res clock 8x TDCLK = 200MHz
    TDHREGEN + // Regulated mode, locked to input clock
    TDHEN;  // Hi-res enable

// Configure the CCRx blocks
TD0CCR0 = 2000;  // PWM Period. Sw freq = 200MHz/2000 = 100 kHz
TD0CCTL1 = OUTMOD_7 + CLLD_1; // CCR1 reset/set mode, updated at start of period
TD0CCR1 = 1000;  // CCR1 PWM duty cycle of 1000/2000 = 50%
TD0CCTL2 = OUTMOD_7 + CLLD_1; // CCR2 reset/set mode, updated at start of period
TD0CCR2 = 500;   // CCR2 PWM duty cycle of 500/2000 = 25%
TD0CTL0 |= MC_1 + TDCLR;  // up-mode, clear TDR, Start timer
Timer D
Control Register TD0CTL0

See *MSP430x5xx/6xx Family User Guide*, Chapter 19, p. 535

C code:
TD0CTL0 = TDSSEL_2;

This sets the Timer D clock source to SMCLK = 25 MHz (derived from processor clock DCO)

TD0CTL0 is a variable associated with this control register in the header file *msp430f5172.h*

TDSSEL_2 is a constant defined in the standard header file, having 01b as bits 9-8. The header file *msp430f5172.h* defines such constants for every control register field.
C code:
TD0HCTL0 = TDHM_0 + TDHREGEN + TDHEN;

This sets the TDHEN bit to enable high resolution mode

The high-res clock is in regulated mode, synchronized to SMCLK

The TDHM bits are set to 0, which causes the hi-res clock to be 8x SMCLK = 8 x 25 MHz = 200 MHz. So each clock count is 5 ns
Timer D
Control Register TD0CCTL1

See MSP430x5xx/6xx Family User Guide, Chapter 19, p. 540

C code:
TD0CCTL1 = OUTMOD_7 + CLLD_1;

This sets the Reset/set mode and update of the duty-cycle command (value in TD0CCR1) when TDR counts to 0, i.e. at the start of a switching period

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Type</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15-14 | CnK | R/W | 6h | Capture mode  
00b = No capture  
01b = Capture on rising edge  
10b = Capture on falling edge  
11b = Capture on both rising and falling edges |
| 13-12 | CCISx | R/W | 6h | Capture/compare input select. These bits select the TDOCCRx input signal. See the device-specific data sheet for specific signal connections.  
00b = CCANx  
01b = CCAN  
10b = CCAD  
11b = VCC |
| 11 | SCS | R/W | 6h | Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. In high-resolution mode, the capture is always synchronous to the high-resolution clock, and this setting is ignored.  
0b = Asynchronous capture  
1b = Synchronous capture |
| 10-9 | CLLDx | R/W | 6h | Compare latch load. These bits select the compare latch load event.  
00b = TDOxLx leads on write to TDOCCRx  
01b = TDOxLx leads when TDR counts to 0  
10b = TDOxLx leads when TDR counts to 0 (up or continuous mode). TDOxLx leads when TDR counts to TDOxCR or to 0 (up/down mode)  
11b = TDOxLx leads when TDR counts to TDOxLx |
| 8 | CAP | R/W | 6h | Capture mode  
0b = Compare mode  
1b = Capture mode |
| 7-6 | OUTMCx | R/W | 6h | Output mode  
00b = OUT bit value  
01b = Set  
10b = Toggle/set  
11b = Set/toggle |
| 4 | CCIE | R/W | 6h | Capture/compare interrupt enable. Enables the interrupt request of the corresponding CnRF bit.  
0b = Interrupt disabled  
1b = Interrupt enabled |
| 3 | CCI | R | 6h | Capture/compare input. The selected input signal can be read by this bit.
Example: Configuring Timer D0 as a PWM with 100 kHz switching frequency
(main.c linked on the Exp 2 page)

// Configure TimerD in Hi-Res Regulated Mode
TD0CTL0 = TDSSEL_2; // TDCLK=SMCLK=25MHz=Hi-Res input clk select
TD0CTL1 |= TDCLKM_1; // Select Hi-res local clock
TD0HCTL1 |= TDHCLKCR; // High-res clock input >15MHz
TD0HCTL0 = TDHM_0 +
TDHREGEN + // Hi-res clock 8x TDCLK = 200MHz
TDHEN; // Hi-res enable

// Configure the CCRx blocks
TD0CCR0 = 2000; // PWM Period. Sw freq = 200MHz/2000 = 100 kHz
TD0CCTL1 = OUTMOD_7 + CLLD_1; // CCR1 reset/set mode, updated at start of period
TD0CCR1 = 1000; // CCR1 PWM duty cycle of 1000/2000 = 50%
TD0CCTL2 = OUTMOD_7 + CLLD_1; // CCR2 reset/set mode, updated at start of period
TD0CCR2 = 500; // CCR2 PWM duty cycle of 500/2000 = 25%
TD0CTL0 |= MC_1 + TDCLR; // up-mode, clear TDR, Start timer

The TD0.1 and TD0.2 outputs will now continue to run at 100 kHz with duty cycles of 0.5
and 0.25. Subsequent writes to TD0CCR1 or TD0CCR2 will cause the output duty cycle to
change at the next 100 kHz switching period.
Development board schematic: header pinout
(complete schematic linked on the Exp 2 page)
Introduction to MSP430
ECEN 4517/5517

- MSP430 microcontroller
- MSP430 development board
- Simple code examples
- Timer D, high-resolution digital pulse-width modulation
- Analog-to-Digital conversion (ADC)
ADC10: The 10-Bit A/D Converter of the MSP430

Key features:
- Multiplexed inputs
- Sample and hold circuit
- Successive approximation register, driven by selectable clock
- Selectable reference sources
- Buffered output memory
- 10 bit or 8 bit conversion
Successive Approximations

- After the input signal has been sampled, the 10-bit SAR requires 11 clock cycles to generate an output.
- Compare analog input with references.
- The MSP430 uses a switched capacitor scheme to perform the comparisons.
- See MSP430x5xx Family User’s Guide, Ch. 27.

Capacitor bypassing is required

**What the User’s Guide recommends:**

Also need capacitance (an RC low-pass filter) at each analog input pin.
Setting up the A/D Converter ADC10

// Configure ADC10
ADC10CTL0 = ADC10SHT_2 + ADC10ON; // sample time of 16 clocks, turn on
    // use internal ADC 5 MHz clock
ADC10CTL1 = ADC10SHP + ADC10CONSEQ_0; // software trigger to start a sample
    // single channel conversion
ADC10CTL2 = ADC10RES; // use full 10 bit resolution
ADC10MCTL0 = ADC10SREF_1+ADC10INCH_5; // ADC10 ref: use VREF and AVSS
    // input channel A5 (pin 10)

// Configure internal reference VREF
while(REFCTL0 & REFGENBUSY); // if ref gen is busy, wait
    REFCTL0 |= REFVSEL_0 + REFON; // select VREF = 1.5 V, turn on
    _delay_cycles(75); // delay for VREF to settle

The above code sets up the 10-bit ADC with A5 as its only input, with 1.5 V
giving a reading of $2^{10} - 1$, and 0 V giving a reading of 0. Each reading will
employ a sampling window of 16 ADC clocks = 3.2 µsec.
Sampling the ADC input

ADC10CTL0 |= ADC10ENC + ADC10SC; // sampling and conversion start
while(ADC10CTL1 & ADC10BUSY); // wait for completion
X = ADC10MEM0; // ADC10MEM0 contains result

The above code is simple and a good start. See CCS5 code examples for use of interrupts that do not require the processor to wait during the conversion time.
Goals in upcoming weeks
Exp. 3: A three-part experiment

Exp. 3 Part 1:
Demonstrate dc-dc converter power stage operating open loop, driven by MSP430 PWM output
Inside, with input power supply and resistive load
Outside, between PV panel and battery
DC system simulation

Exp. 3 Parts 2 and 3:
Demonstrate working sensor circuitry, interfaced to microprocessor
Demonstrate peak power tracker and battery charge controller algorithms, outside with converter connected between PV panel and battery
Converter Power Stage
Some choices

Buck converter
- Steps down voltage
- Industry workhorse
- High efficiency

SEPIC
- Can step voltage up or down, to peak power track over wider voltage range
- More complex
- Good efficiency
Exp. 3, Part 1
Demonstrate dc-dc power stage inside
Prelab assignment
Exp. 3, Part 1

Design your buck converter power stage

1. Work out the current waveforms of each component: MOSFET, diode, inductor, capacitors

2. Design your inductor
   • Use $K_g$ method explained in ECEN 4797/5797
   • You decide how much ripple to allow, how much power loss to allow, etc.
   • Use one of the ferrite cores in your kit

3. Check the voltage and current stresses on each power component and make sure the components operate within their datasheet ratings

Contents of parts kit, with links to datasheets, is on web at

http://ece.colorado.edu/~ecn4517/components/kit.html
Component stresses

Sketch the waveform for each power component and choose components that operate within their datasheet specified ratings:

MOSFETs: peak voltage, average current
Diodes: peak inverse voltage, average current
Capacitors: maximum working voltage, rms current

Most companies will apply derating factors to many of the datasheet limits

Example: 60% to 80% of datasheet max voltage value is allowed for power semiconductors. This value includes worst-case peak transient voltages.
Core Material 7070
TSC Ferrite International

See parts kit web page for complete datasheets

Kit includes ferrite cores made of this material, in three geometries:
PQ 32/20
PQ 26/25
13-07-06 toroid
Gate drive circuit
with transformer isolation

- Gate driver output $v_d(t)$ has a dc component when $d \neq 0.5$
- Transformer will saturate if we apply dc
- Primary blocking capacitor removes dc component
- Secondary capacitor and diodes form a diode clamp circuit that restores the dc component
Gate driver transformer

- Use ferrite toroid in your kit
- Leakage inductance is minimized if bifilar winding is used
- Need enough turns so that applied volt-seconds do not saturate core:

$$\Delta B = \frac{V_1 DT_s}{n_1 A_c}$$
Alternate smaller version of gate driver

- Uses only one gate driver instead of two, to produce half the voltage swing on primary
- Transformer turns ratio is 1:1
- Produces half as much gate current
- Suitable for smaller MOSFETs
Exp. 3, Part 1
Test open-loop converter, outside

Basic control characteristics:
How does the duty cycle control the PV and battery voltages and currents?
Layout issues

Example: Buck converter

Use loop analysis

Switched input current $i_1(t)$ contains large high frequency harmonics
—hence inductance of input loop is critical
inductance causes ringing, voltage spikes, switching loss, generation of B- and E-fields, radiated EMI

The second loop contains a filter inductor, and hence its current $i_2(t)$ is nearly dc
—hence additional inductance is not a significant problem in the second loop
Parasitic inductances of input loop explicitly shown:

Addition of bypass capacitor confines the pulsating current to a smaller loop:

High frequency currents are shunted through capacitor instead of input source.
Even better: minimize area of the high frequency loop, thereby minimizing its inductance
Example: gate driver

line input

+15 volt supply

analog control chip

PWM control chip

gate driver

power MOSFET

\[ i_g(t) \]

Department of Electrical and Computer Engineering
University of Colorado at Boulder
Solution: bypass capacitor and close coupling of gate and return leads

High frequency components of gate drive current are confined to a small loop

A dc component of current is still drawn output of 15V supply, and flows past the control chips. Hence, return conductor size must be sufficiently large