ECEN 4517/5517

Power Electronics and Photovoltaic Power Systems Laboratory

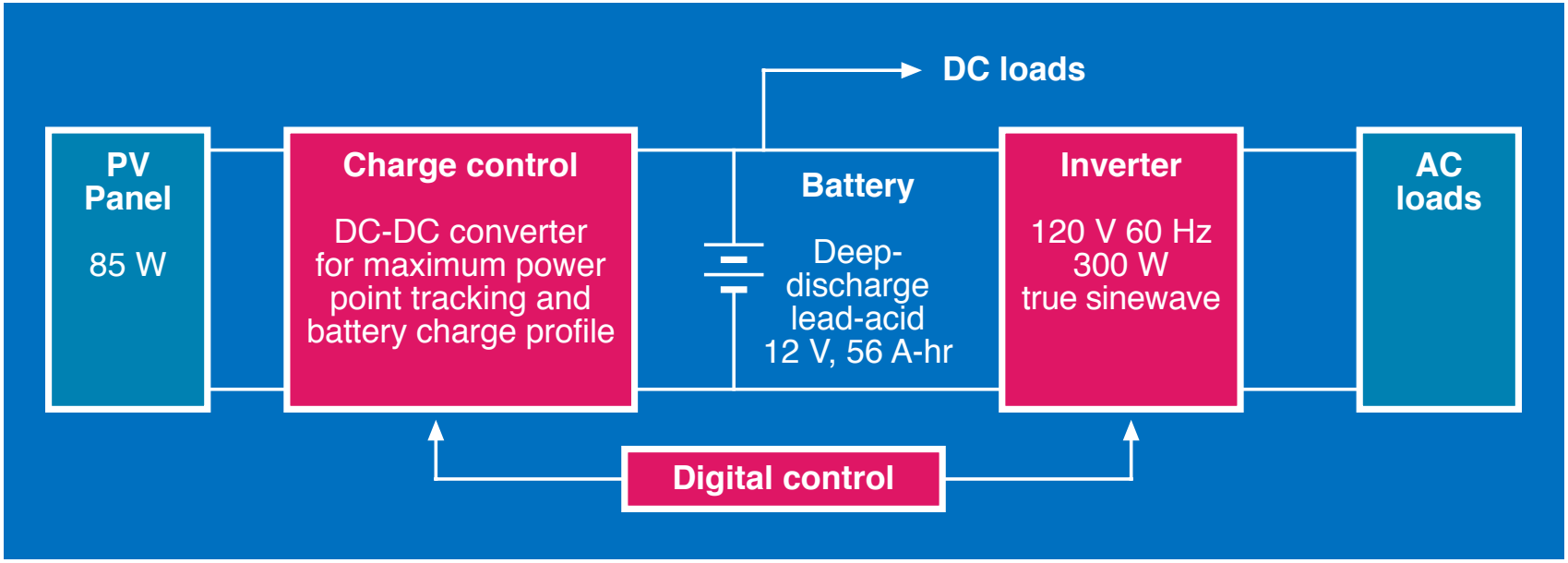
Lecture 2

TI MSP430 Microcontroller

Announcements

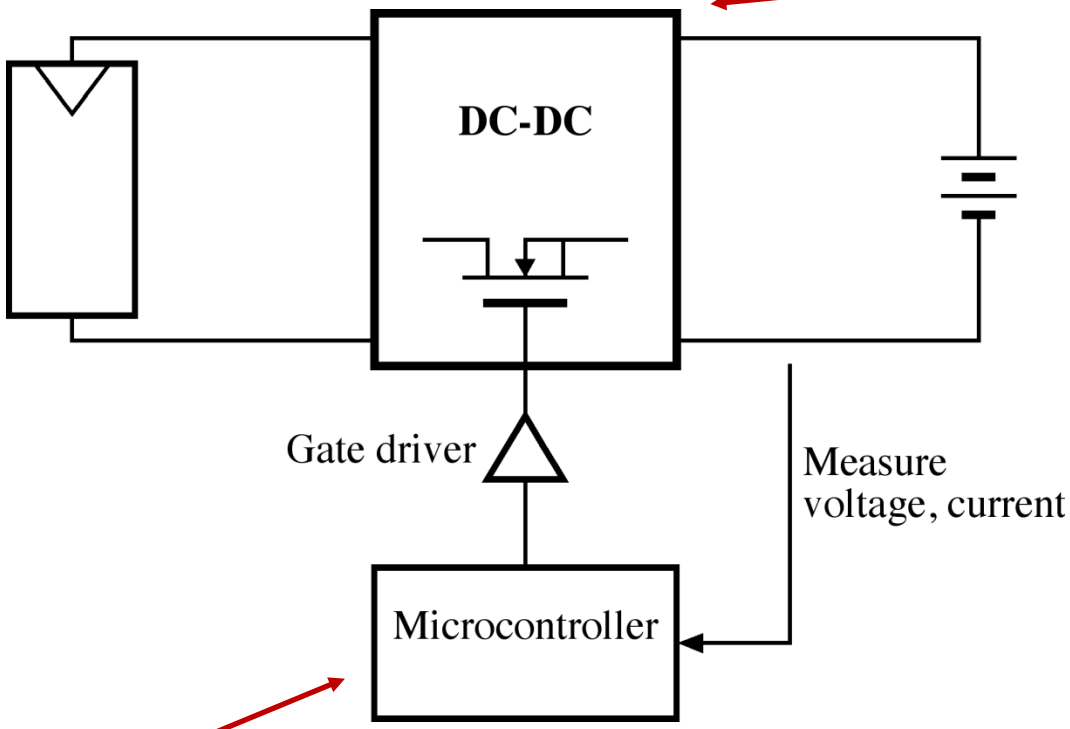
- Experiment 1 Lab Report due by 11:59 pm (MT) on Friday January 27, 2017
- Experiment 3-1 Pre-lab Assignment is out
  - Due by 11:59 pm (Mountain Time) on Friday January 27, 2017
- This week's lab: Experiment 2 (Need Parts Kit for this)
  - Lab Assignment and related documents posted on D2L
  - Experiment 2 has no pre-lab
  - Exp 2 Lab Report Sheet due by 11:59 pm (MT) on Friday February 3, 2017

Experiments



- Exp 1 – PV panel and battery characteristics and direct energy transfer
- Exp 2 – TI MSP430 microcontroller introduction
- Exp 3-1, 3-2 – Buck dc-dc converter for PV MPPT and battery charge control
- Exp 4 – Step-up 12V-200V dc-dc converter
- Exp 5 – Single-phase dc-ac converter (inverter)
- Expo – Complete system demonstration

Experiments 2 and 3



Exp 3: DC-DC Converter

- Exp 3-1: Design and construct dc-dc converter
- Exp 3-2: Employ microcontroller to achieve MPPT and battery charge control

Exp 2: Introduction to MSP430 Microcontroller

- Learn to use TI MSP430 microcontroller
- Set up your MSP430 to drive a MOSFET at a programmable duty cycle

MSP430F5172 Microcontroller Programming

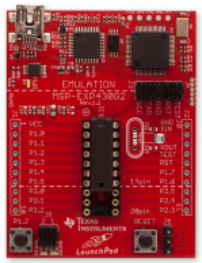
- Code Composer Studio 6.1.3
  - Development system for MSP430; program in C
  - On lab computers: free version (limited code size)

MSP430F5172 Development Board

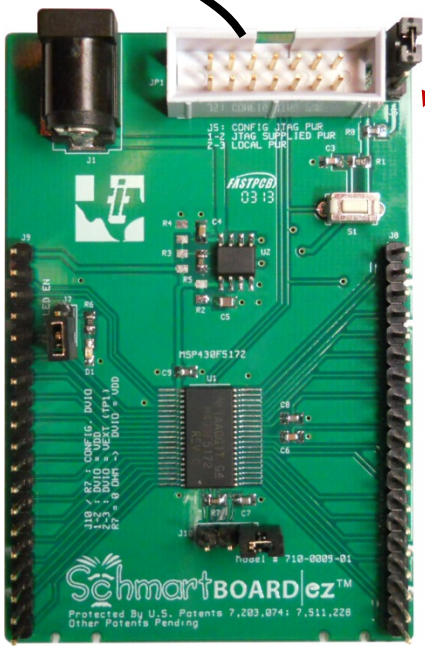
Programming and Debugging Interface

MSP430 Launchpad

OR



MSP430 USB-Debug Interface (also called JTAG Programmer)



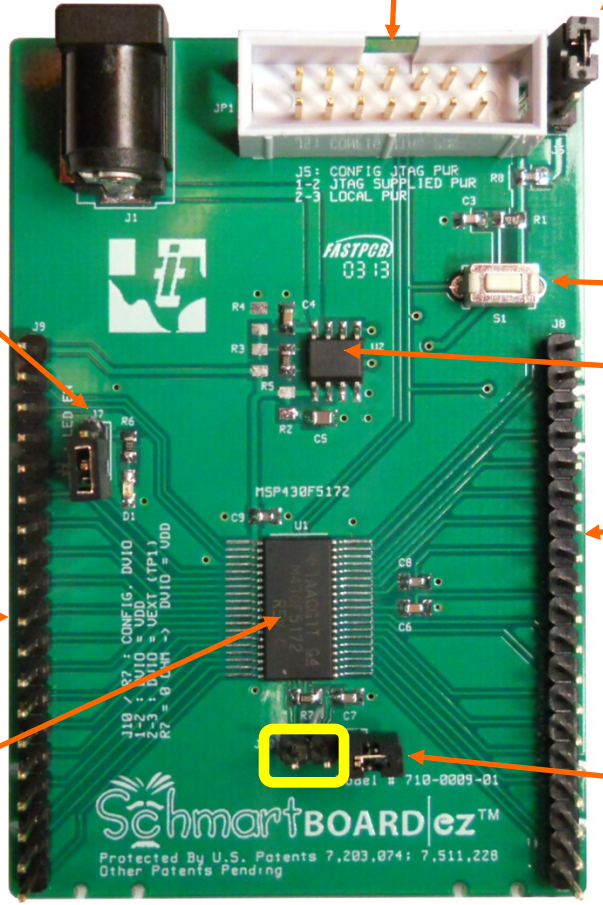
MSP430F5172 Learning Resources

- MSP430F5172 User's Guide
  - The primary resource for operation and programming of on-chip peripherals (PWM, ADC, etc.) - 1147 pages
- MSP430F5172 Datasheet
  - Describes pinouts, specifications – 103 pages
- Library of Code Examples
  - Accessible within Code Composer Studio, also on D2L website
  - Many programming examples for each peripheral
  - Use directory of examples for 430F5172 chip
  - Most useful: Erickson's sample code main.c on D2L website

Main.c

```
#include <msp430.h>

void SetVcoreUp (unsigned int level);

/*
 * main.c
 */
void main(void) {
    volatile unsigned long i;   // Declare counter variable
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    P1SEL |= BIT6;                          // Set P1.6 to output direction (Timer D0.0 output)
    P1DIR |= BIT6;
    P1SEL |= BIT7;                          // Set P1.7 to output direction (Timer D0.1 output)
    P1DIR |= BIT7;
    P2SEL |= BIT0;                          // Set P2.0 to output direction (Timer D0.2 output)
    P2DIR |= BIT0;
    P1DIR |= 0x01;                          // Set P1.0 to output direction (to drive LED)
    P1OUT |= 0x01;                          // Set P1.0  - turn LED on
    __delay_cycles(500000);
    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-or function  - turn LED off

    // 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 5xx
    // User Guide for optimization.
    // 32 x 32 x 25 MHz / 32,768 Hz = 782000 = MCLK cycles for DCO to settle
    __delay_cycles(782000);

     // 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
    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

    // Wait some, allow hi-res clock to lock
    P1OUT ^= 0x01;                                                  // Toggle P1.0 using exclusive-OR, turn LED on
    // __delay_cycles(500000);
    while(!TDHLKIFG);                                       // Wait until hi-res clock is locked
    P1OUT ^= 0x01;                                                  // Toggle P1.0 using exclusive-OR, turn LED off

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

    for (;;) {                                              // Infinite loop, blink LED
                P1OUT ^= 0x01;              // Toggle P1.0 output
                i = 1000000;
                do(i--);
                while(i != 0);              // Wait 10000 cycles
        }
}

  void SetVcoreUp (unsigned int level)
  {
       // Subroutine to change core voltage
    // Open PMM registers for write
    PMMCTL0_H = PMMPW_H;
    // Set SVS/SVM high side new level
    SVSMHCTL = SVSHE + SVSHRVL0 * level + SVMHE + SVSMHRRL0 * level;
    // Set SVM low side to new level
    SVSMLCTL = SVSLE + SVMLE + SVSMLRRL0 * level;
    // Wait till SVM is settled
    while ((PMMIFG & SVSMLDLYIFG) == 0);
    // Clear already set flags
    PMMIFG &= ~(SVMLVLRIFG + SVMLIFG);
    // Set VCore to new level
    PMMCTL0_L = PMMCOREV0 * level;
    // Wait till new level reached
    if ((PMMIFG & SVMLIFG))
      while ((PMMIFG & SVMLVLRIFG) == 0);
    // Set SVS/SVM low side to new level
    SVSMLCTL = SVSLE + SVSLRVL0 * level + SVMLE + SVSMLRRL0 * level;
    // Lock PMM registers for write access
    PMMCTL0_H = 0x00;
}
```

MSP430F5172 Development Board



External Power

JTAG
(to computer)

Jumper
Select power source: JTAG (as shown) or external

Jumper
Connect or disconnect LED from P1.0

Reset Button

Linear Regulator (3.3 V)

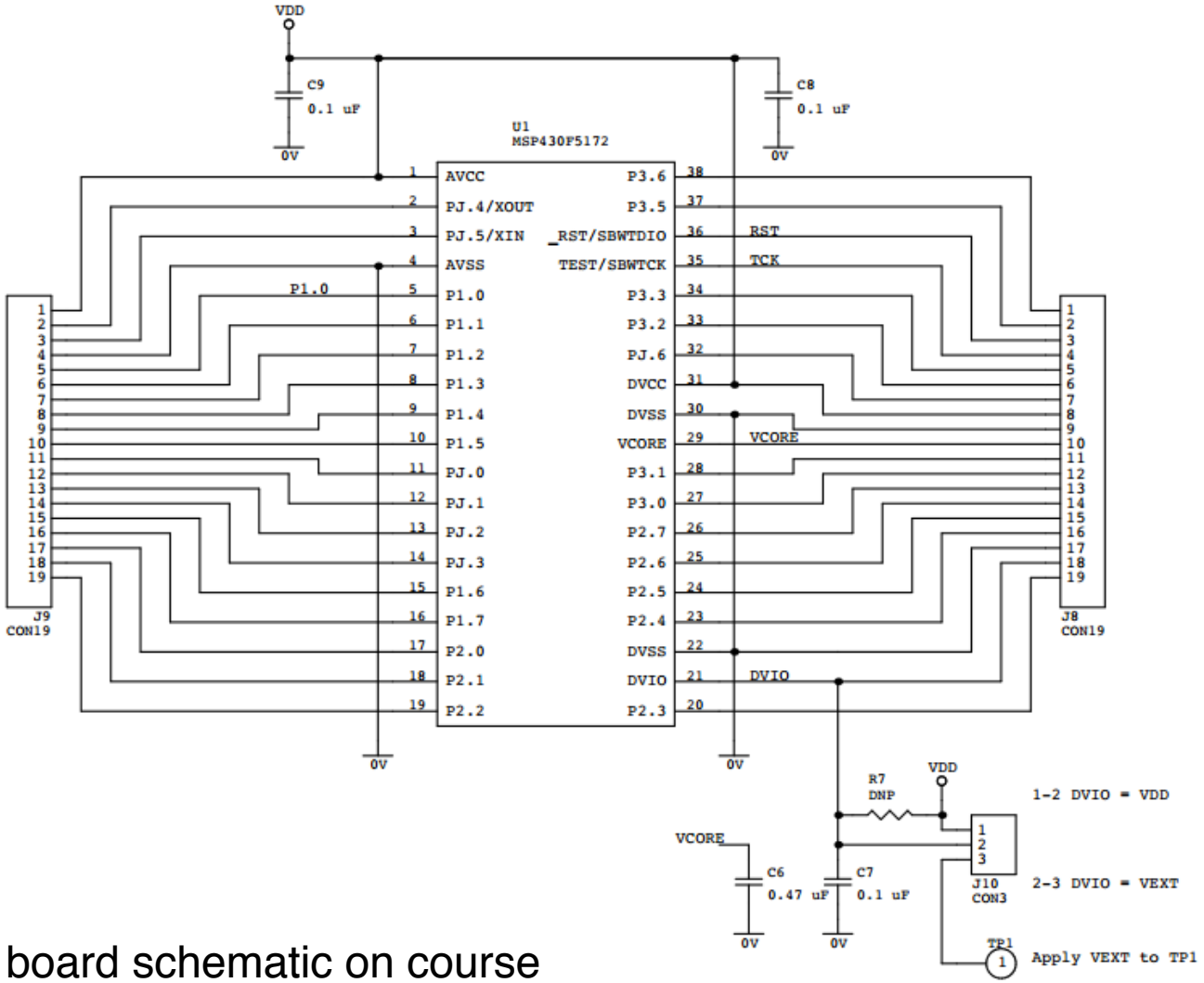
Header
Processor pins 20-38

Header
Processor pins 1-19

Jumper
Select digital I/O power: Internal 3.3 V (place jumper to short 1-2, as highlighted), or external 5 V

MSP430F5172 Microcontroller

Development Board Header Pinout



Complete board schematic on course website under Experiment 2

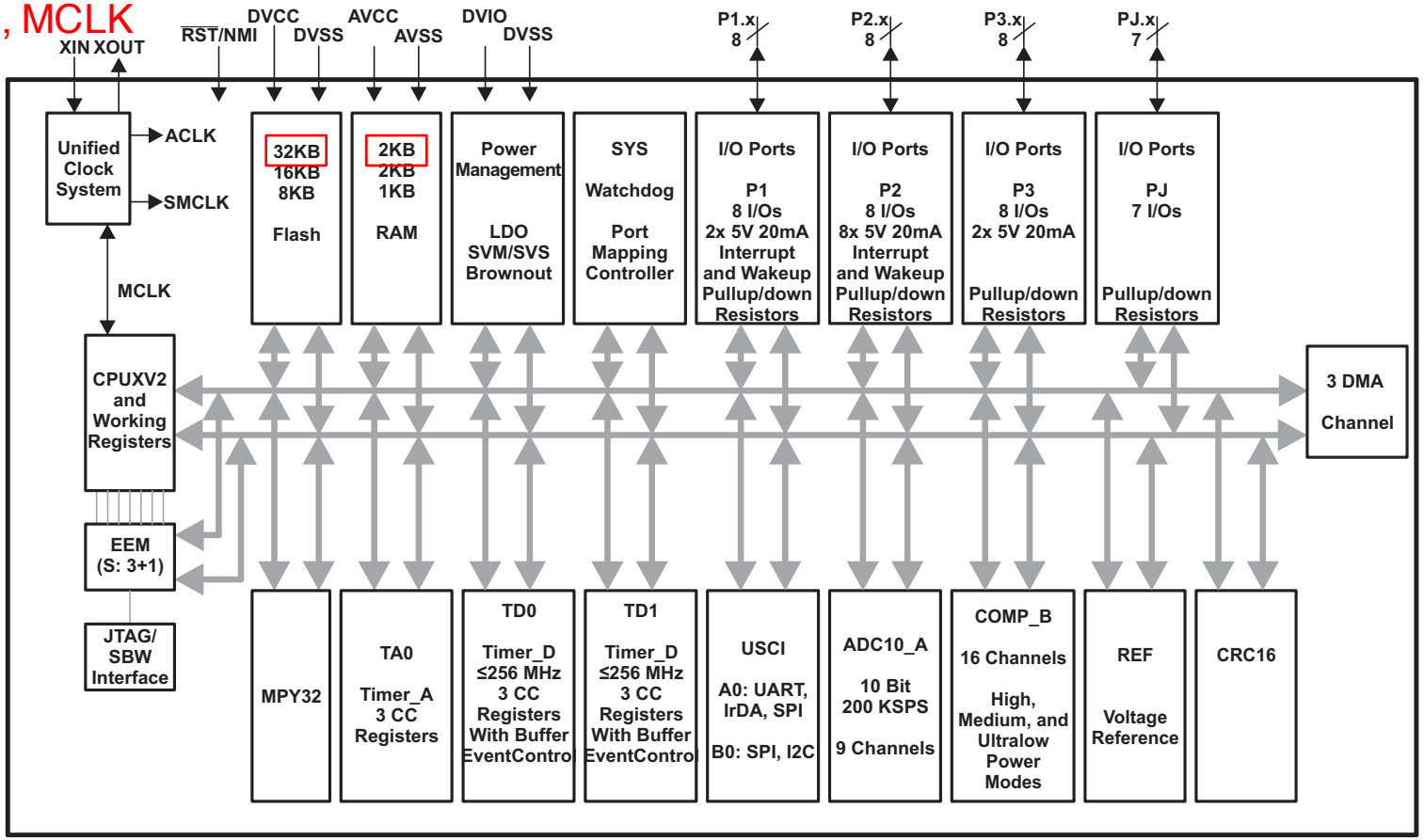
MSP430F5172 Microcontroller

Clocks: ACLK, SMCLK, MCLK

Up to 25 MHz @3.3V

CPU: 16 bit

Programmable multi-use I/O ports (31)



32-bit multiplier

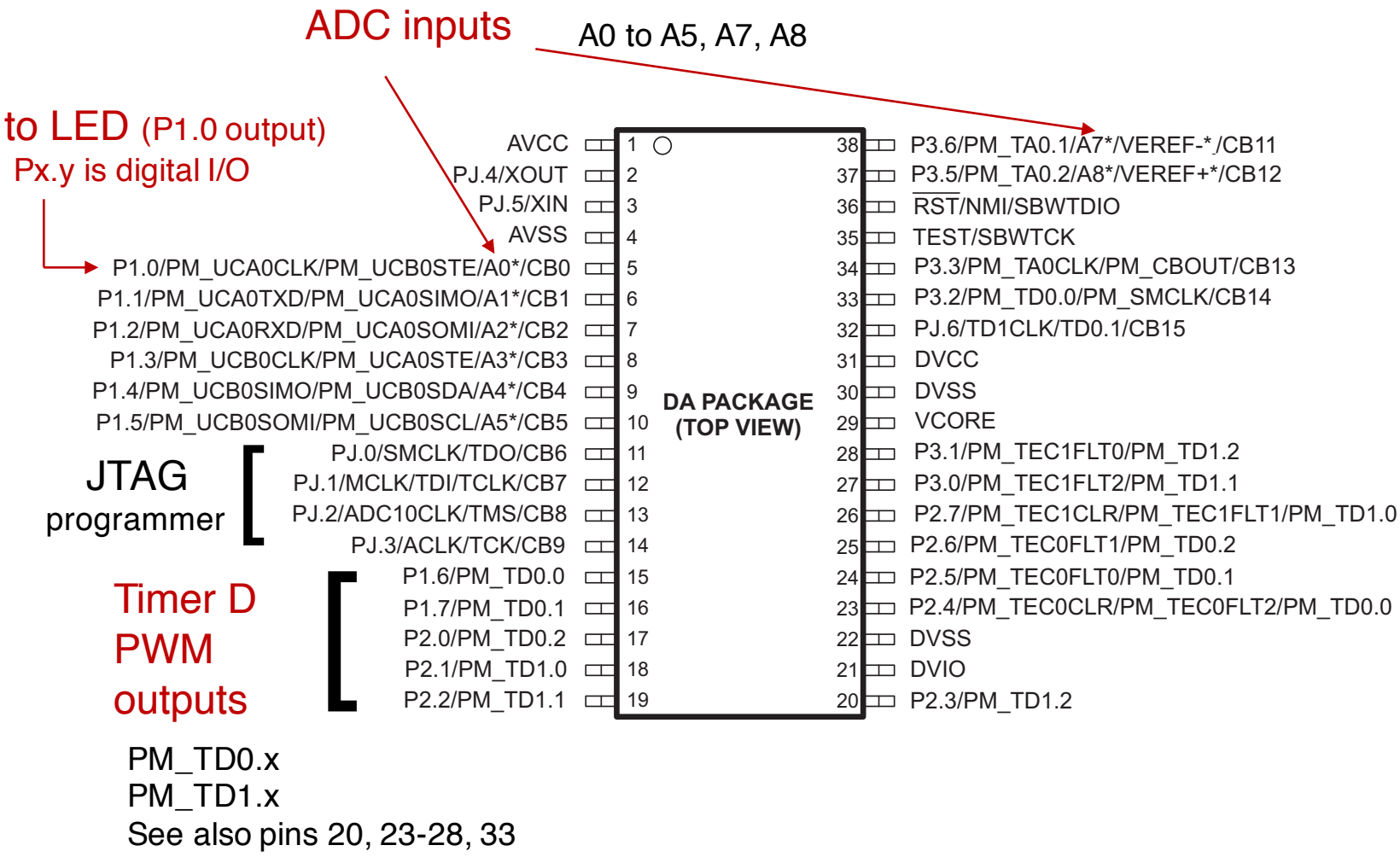
Timer D (2) Timer A (1)

10-bit A/D

Analog comparator

Voltage reference

Microcontroller Pinout



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

Microcontroller Default Settings

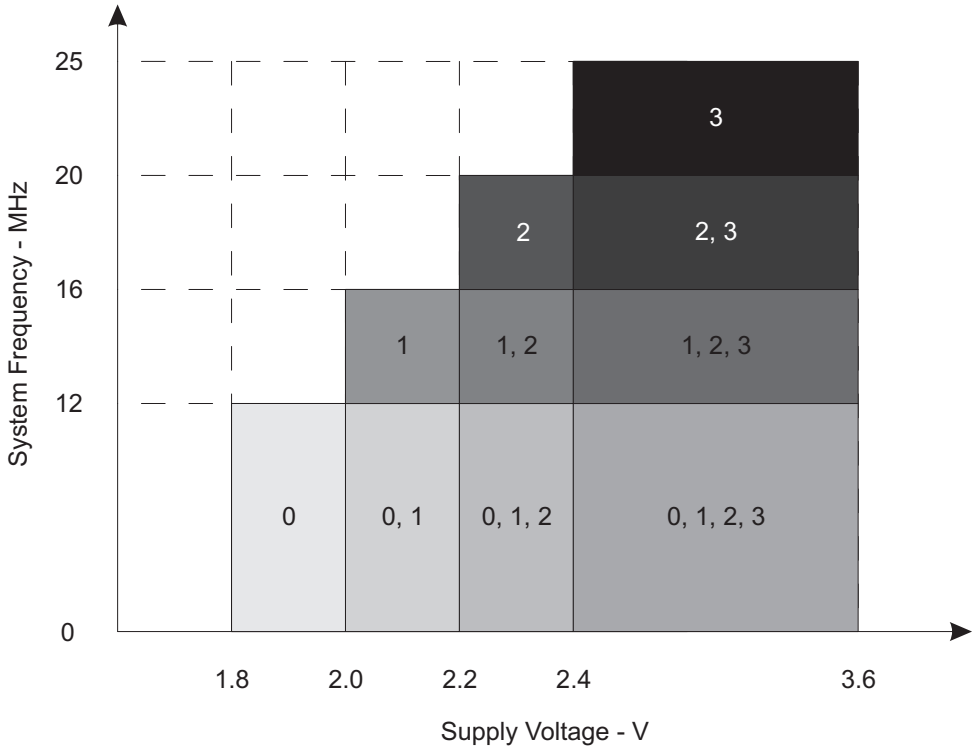
- Microcontroller default settings - upon power-on reset (POR)
  - Watchdog timer is enabled (to turn it off see C code below)
  - All pins are set to read state
  - Processor internal clock and core voltage are set to minimum values; default clock frequency = 1.045 MHz
  - Processor supply voltage is 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.

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
```

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 to Change Clock Frequency

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);
```

Controlling the Ports

- Peripherals are controlled by registers in addressable memory
- 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

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

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

There are additional P1 registers related to interrupts.
For further documentation, see *MSP430x5xx/6xx Family User Guide,* Ch 12, pp. 406

Sample Code for Controlling the Ports

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;          // XOR the contents of P1OUT with hex 01,
                        // toggling the first bit
                        // This changes the state of logic output P1.0
```

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0

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

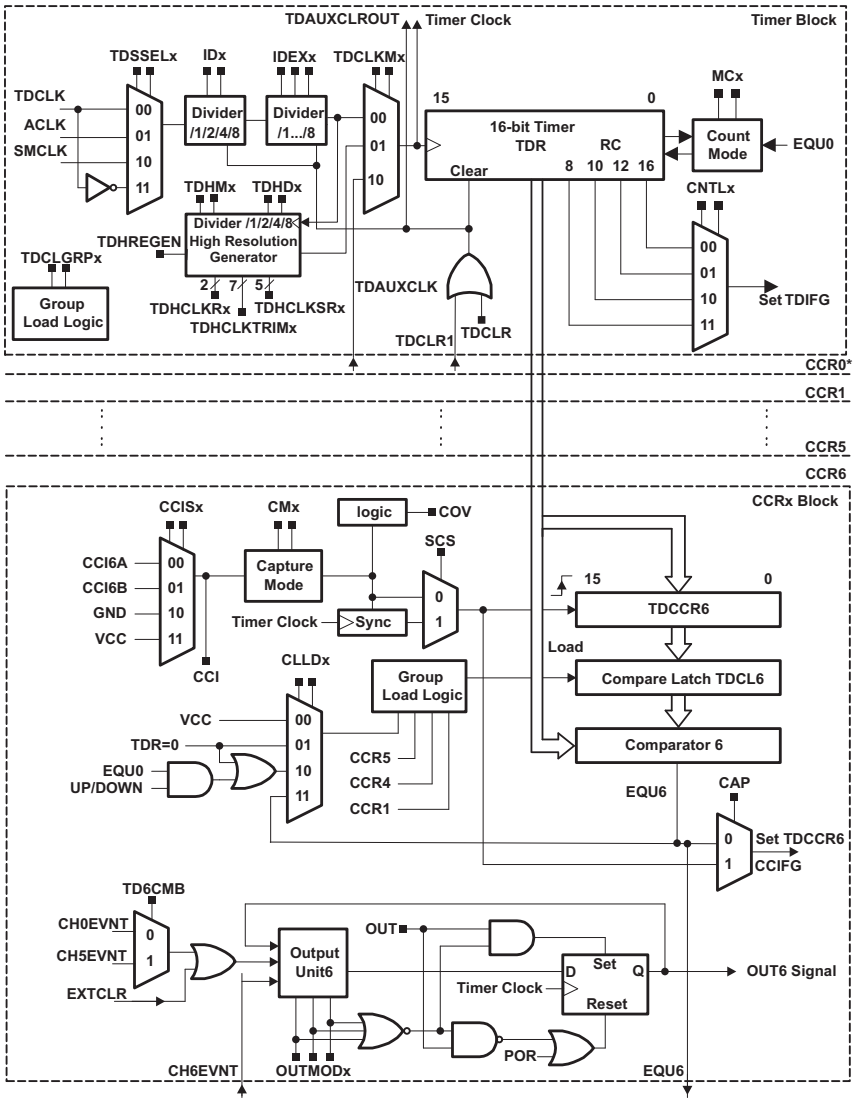
Sample Code to Toggle Pin P2.2

```
#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
        }
}
```

Above code example drives 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 2.

Timer D

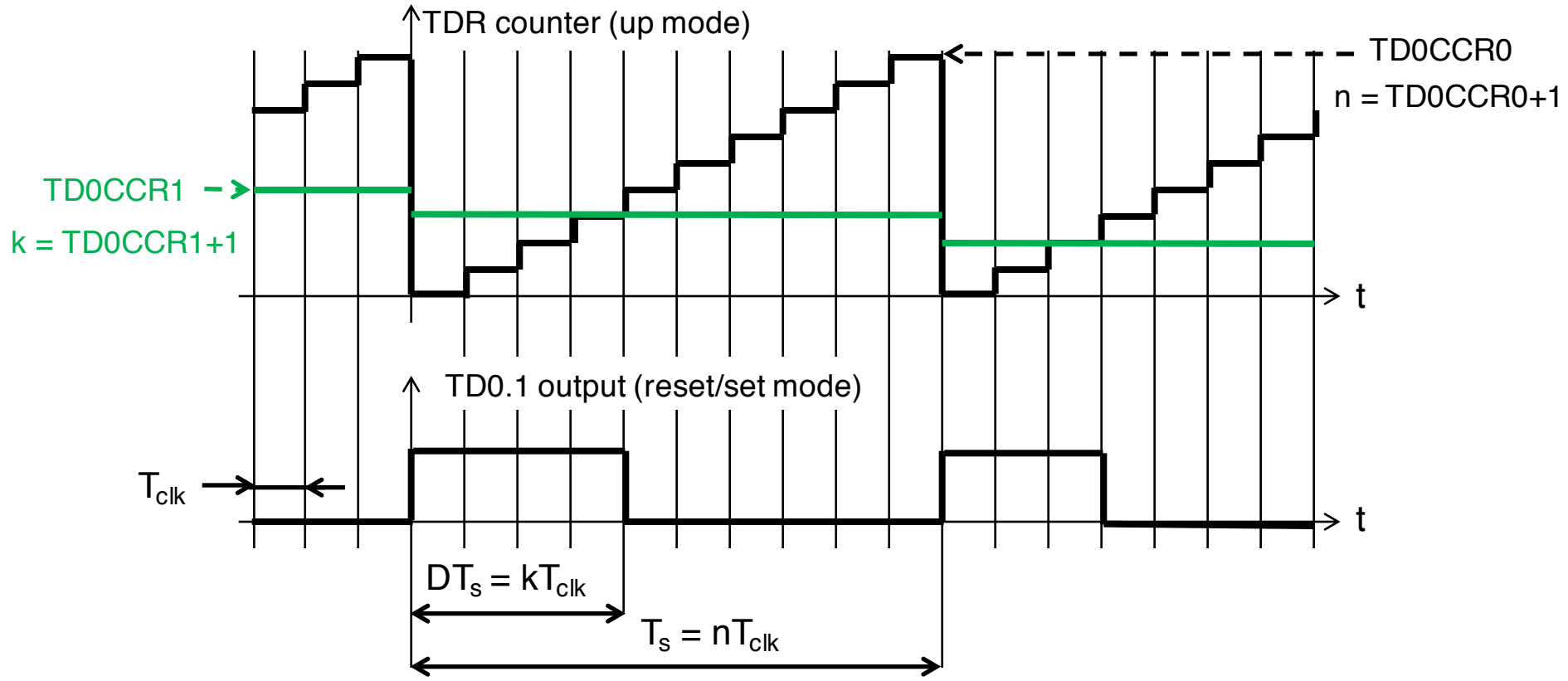
- The MSP430F5172 has two Timer D's
- Each Timer D includes:
  - One timer block with 16 bit counter
  - Three capture/compare registers (CCR0 – CCR2)
  - High resolution mode with TDCLK frequency = N*(DCO frequency)
- Use CCR0 to set switching frequency: fs = (TDCLK freq)/(CCR0)
- Use CCR1 and CCR2 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)



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

PWM Using Timer D

- Peripheral used to generate high-resolution digital pulse-width modulation



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

- Setup core voltage and core clock frequency (SMCLK) to the highest values:
  - Level 3 core voltage = 3.3 V
  - SMCLK frequency = 25 MHz
- Setup high-resolution Timer D generator to multiply SMCLK by 8 to TDCLK frequency = 200 MHz (5 ns time resolution)
- Configure Timer D modes, switching period

High Resolution Timer D PWM Setup

```
// 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
```

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.

Timer D – Control Register TD0CTL0

C code (Line 1):
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.

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

Figure 19-24. TDxCTL0 Register

15	14	13	12	11	10	9	8
Reserved	TDCLGRPx		CNTLx		Reserved	TDSSELx	
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MCx		Reserved	TDCLR	TDIE	TDIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0	w-(0)	rw-(0)	rw-(0)

Table 19-9. TDxCTL0 Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-13	TDCLGRPx	RW	0h	TDCLx group 00b = Each TDCLx latch loads independently. 01b = TDxCL1+TDxCL2 (TDxCCR1 CLLDx bits control the update) TDxCL3+TDxCL4 (TDxCCR3 CLLDx bits control the update) TDxCL5+TDxCL6 (TDxCCR5 CLLDx bits control the update) TDxCL0 independent 10b = TDxCL1+TDxCL2+TDxCL3 (TDxCCR1 CLLDx bits control the update) TDxCL4+TDxCL5+TDxCL6 (TDxCCR4 CLLDx bits control the update) TDxCL0 independent 11b = TDxCL0+TDxCL1+TDxCL2+TDxCL3+TDxCL4+TDxCL5+TDxCL6 (TDxCCR1 CLLDx bits control the update)
12-11	CNTLx	RW	0h	Counter length 00b = 16-bit, TDR(max) = 0FFFFh 01b = 12-bit, TDR(max) = 0FFFh 10b = 10-bit, TDR(max) = 03FFh 11b = 8-bit, TDR(max) = 0FFh
10	Reserved	R	0h	Reserved. Always reads as 0.
9-8	TDSSELx	RW	0h	Timer_D clock source select 00b = TDCLK 01b = ACLK 10b = SMCLK 11b = Inverted TDCLK
7-6	ID	RW	0h	Input divider. These bits, along with the IDEX bits in TDxCTL1, select the divider for the input clock. 00b = Divide by 1 01b = Divide by 2 10b = Divide by 4 11b = Divide by 8
5-4	MCx	RW	0h	Mode control. Setting MCx = 00h when Timer_D is not in use saves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TDCL0 10b = Continuous mode: Timer counts up to the value set by CNTLx (counter length) 11b = Up/down mode: Timer counts up to TDCL0 and down to 0000h
3	Reserved	R	0h	Reserved. Always reads as 0.
2	TDCLR	W	0h	Timer_D clear. Setting this bit resets TDR, the TDCLK divider, and the count direction. The TDCLR bit always read as zero.

Timer D – Control Register TD0HCTL0

C code (Line 4):
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

See *MSP430x5xx/6xx Family User Guide,* Chapter 19, p. 543

Figure 19-31. TDxHCTL0 Register

15	14	13	12	11	10	9	8
Reserved							TDHFW
r0	r0	r0	r0	r0	r0	r0	rw-(0)
7	6	5	4	3	2	1	0
TDHDx		TDHMx		TDHRON	TDHEAEN	TDHREGEN	TDHEN
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 19-16. TDxHCTL0 Register Description

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved. Always reads as 0.
8	TDHFW	RW	0h	High-resolution generator fast wakeup enable 0b = High-resolution generator fast wakeup disabled 1b = High-resolution generator fast wakeup enable
7-6	TDHDx	RW	0h	High-resolution clock divider. These bits select the divider for the high resolution clock. 00b = Divide by 1 01b = Divide by 2 10b = Divide by 4 11b = Divide by 8
5-4	TDHMx	RW	0h	Timer_D high-resolution clock multiplication factor 00b = High-resolution clock 8x Timer_D clock 01b = High-resolution clock 16x Timer_D clock 10b = Reserved 11b = Reserved
3	TDHRON	RW	0h	Timer_D high-resolution generator forced on. 0b = High-resolution generator is on if the Timer_D counter MCx bits are 01, 10 or 11. 1b = High-resolution generator is on in all Timer_D MCx modes. The PMM remains in high-current mode.
2	TDHEAEN	RW	0h	Timer_D high-resolution clock enhanced accuracy enable bit. Setting this bit reduces the accumulated frequency offset of the high-resolution clock generator and the reference clock. 0b = Normal accuracy 1b = Enhanced accuracy enable
1	TDHREGEN	RW	0h	Timer_D regulation enable. Set this bit to synchronize the high-resolution clock to the Timer_D input clock defined by TDSSELx. 0b = Regulation disabled 1b = Regulation enabled
0	TDHEN	RW	0h	Timer_D high-resolution enable bit. This bit must be set to enable high-resolution operation mode. Whenever a high-resolution TDAUXCLK from another Timer_D instance is used, this bit must also be set. 0b = High-resolution mode disable 1b = High-resolution mode enable

Timer D – Control Register TD0CCTL1

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

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

Table 19-13. TDxCCTLn Register Description

Bit	Field	Type	Reset	Description
15-14	CMx	RW	0h	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	RW	0h	Capture/compare input select. These bits select the TDCCRx input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	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	RW	0h	Compare latch load. These bits select the compare latch load event. 00b = TDCLx loads on write to TDCCRx 01b = TDCLx loads when TDR counts to 0 10b = TDCLx loads when TDR counts to 0 (up or continuous mode). TDCLx loads when TDR counts to TDxCL0 or to 0 (up/down mode). 11b = TDCLx loads when TDR counts to TDCLx
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMODx	RW	0h	Output mode 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. Enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.

PWM Related Pins