CS4101 Introduction to Embedded Systems

Lab 3: Timer and Clock

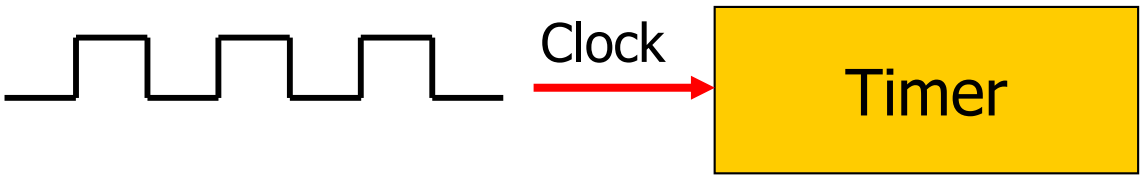
Prof. Chung-Ta King

Department of Computer Science

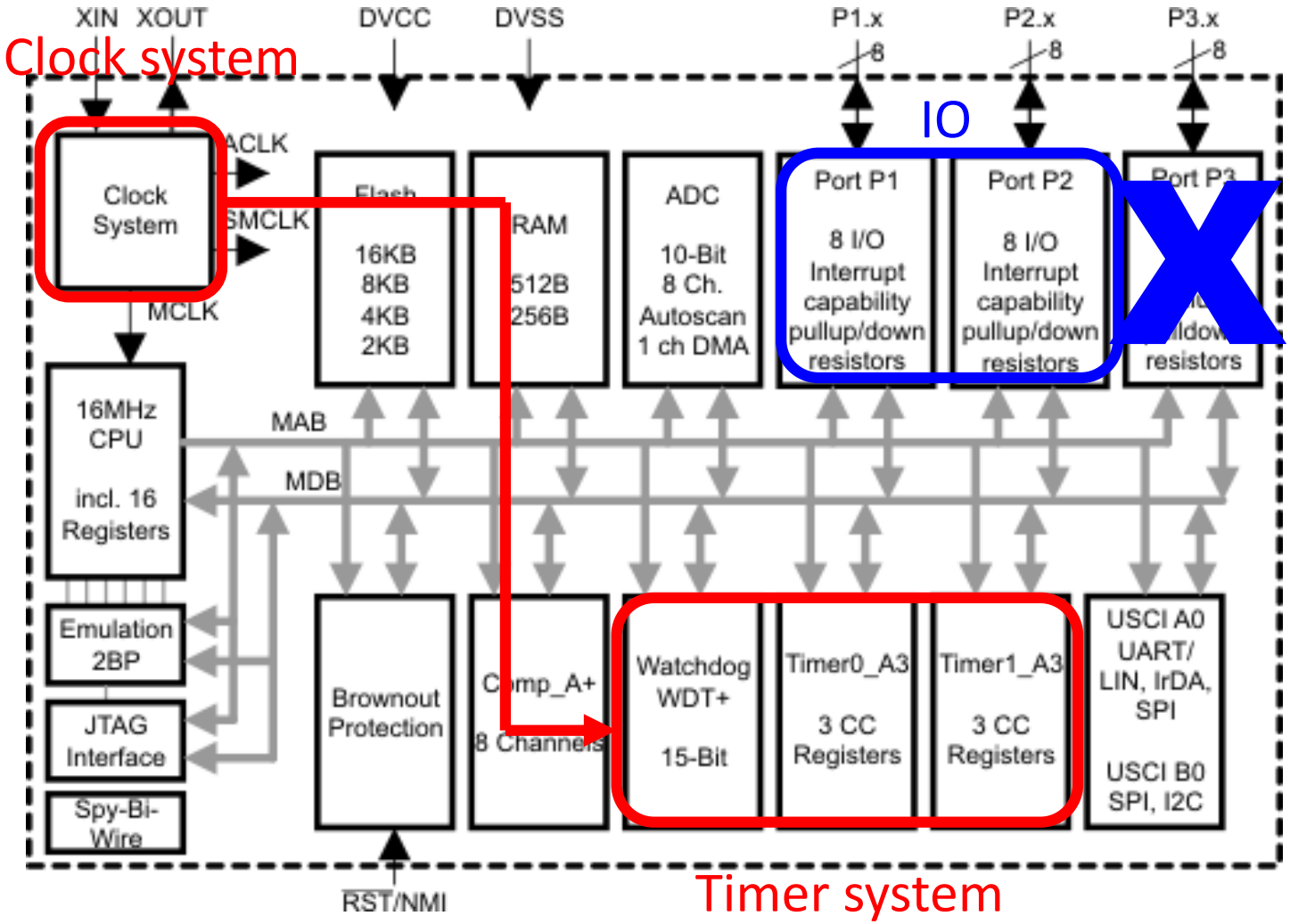
National Tsing Hua University, Taiwan

Introduction

- In this lab, we will learn more advanced timer operations and clocking of MSP430 LanuchPad
  - Capture/compare block of the timer
  - Characteristics of different clock sources and their settings



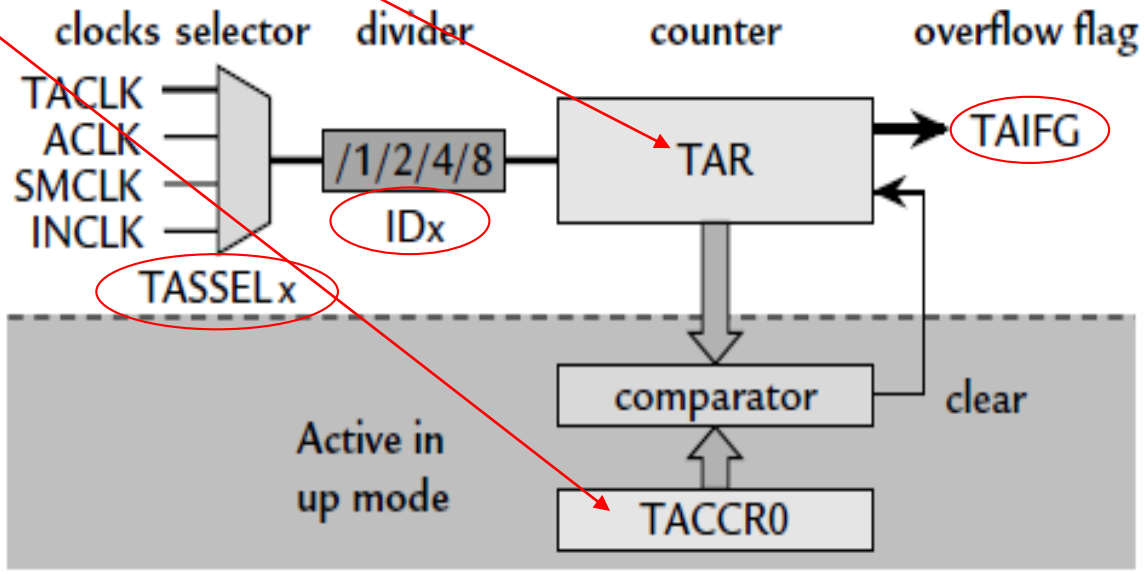
Interior of MSP430G2553



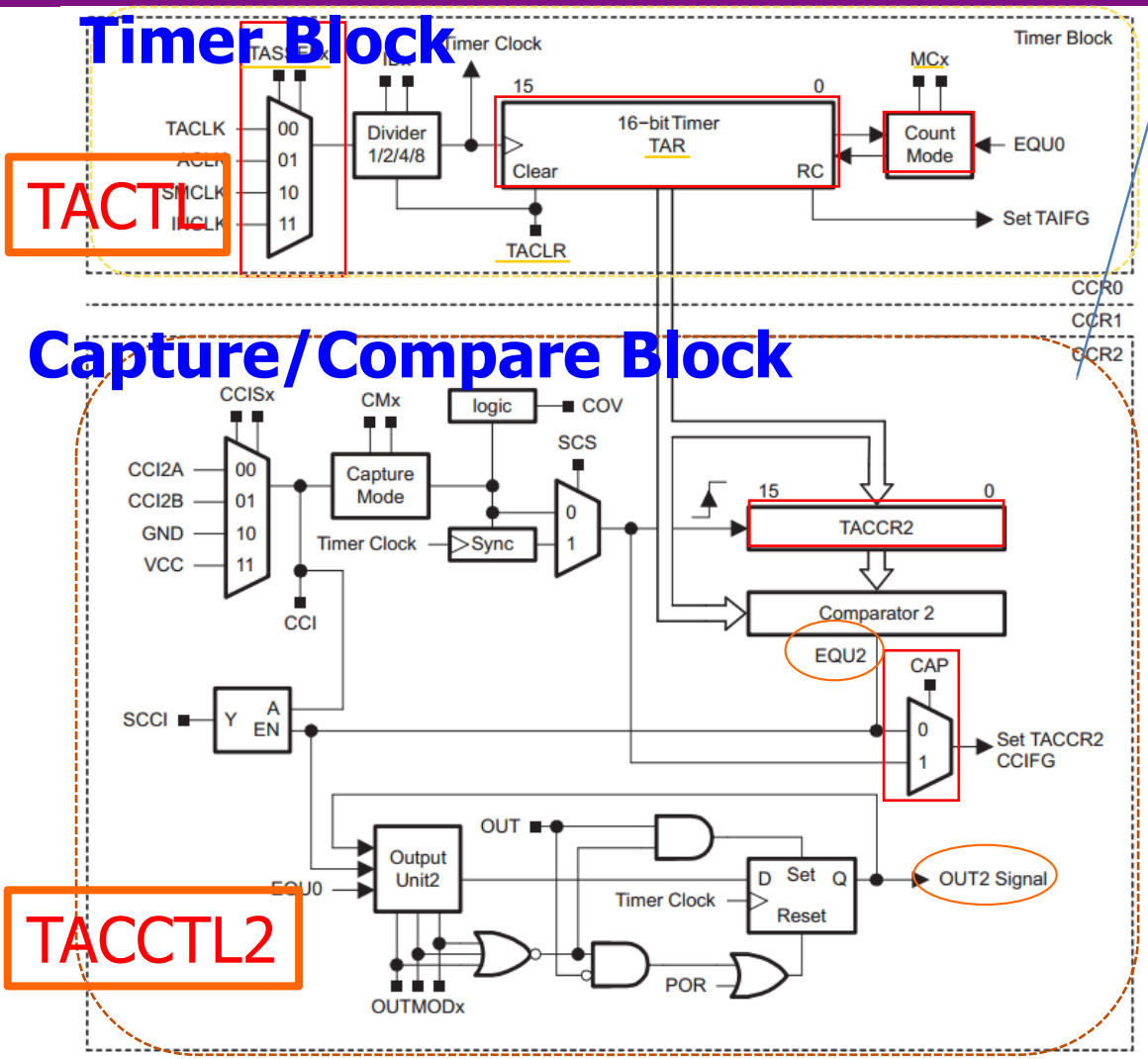
Not available on 20-pin device

Recall: MSP430 Timer_A

- TAR (0170h): the counter itself
- TACTL (0160h): register to control TAR settings
- TACCR0 (0172h): target for counting
- Others: clock source selection, flags



Timer_A Capture/Compare Block

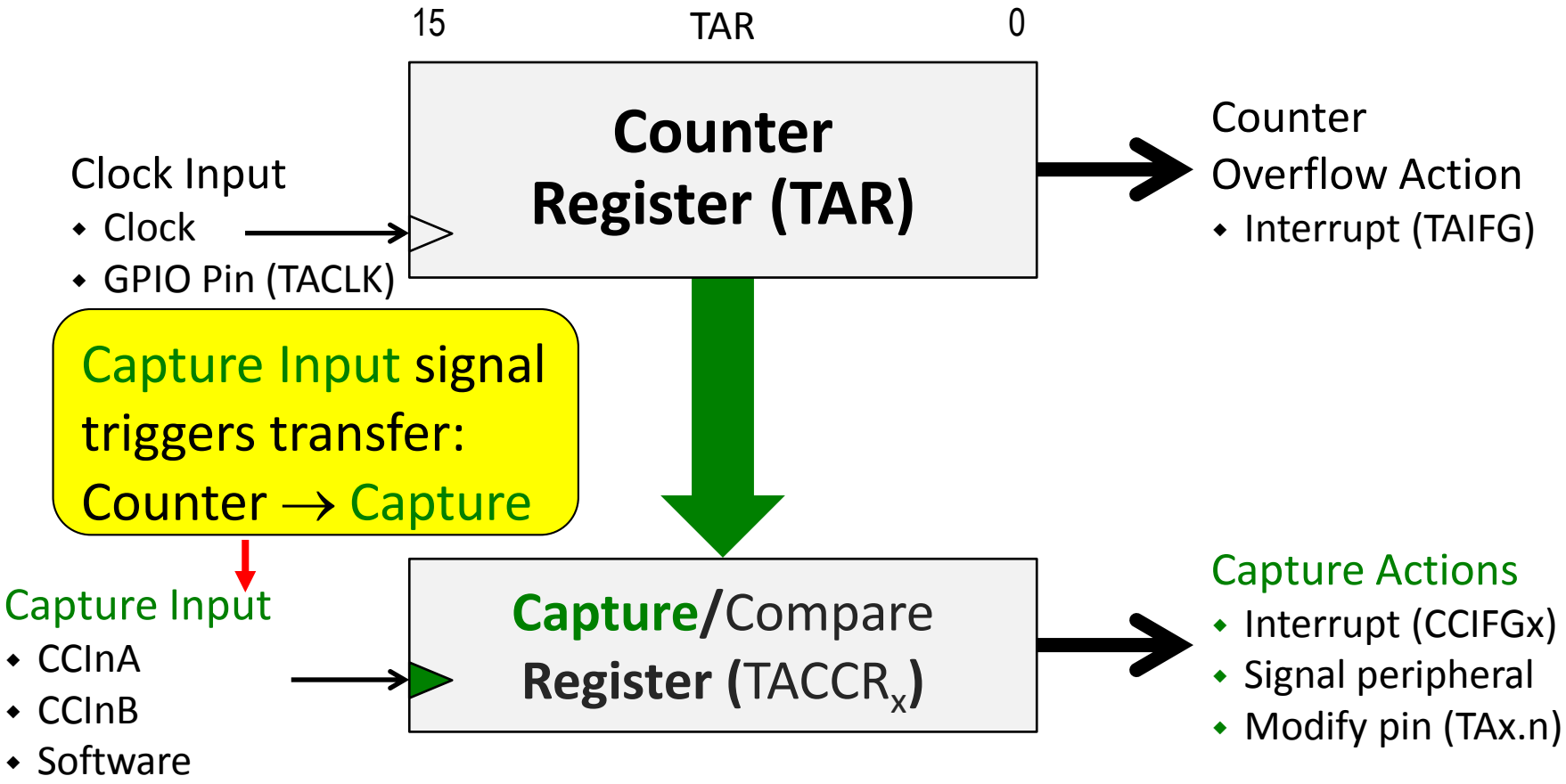


- May contain several Capture/Compare Blocks
- Each C/C block is controlled by a control register, **TACCTLx**
- Inside each C/C block, the Capture/Compare Register, **TACCRx**, holds the count to configure timer
- But, all C/C blocks within Timer_A share the same timer block: TAR

Modes of Capture/Compare Block

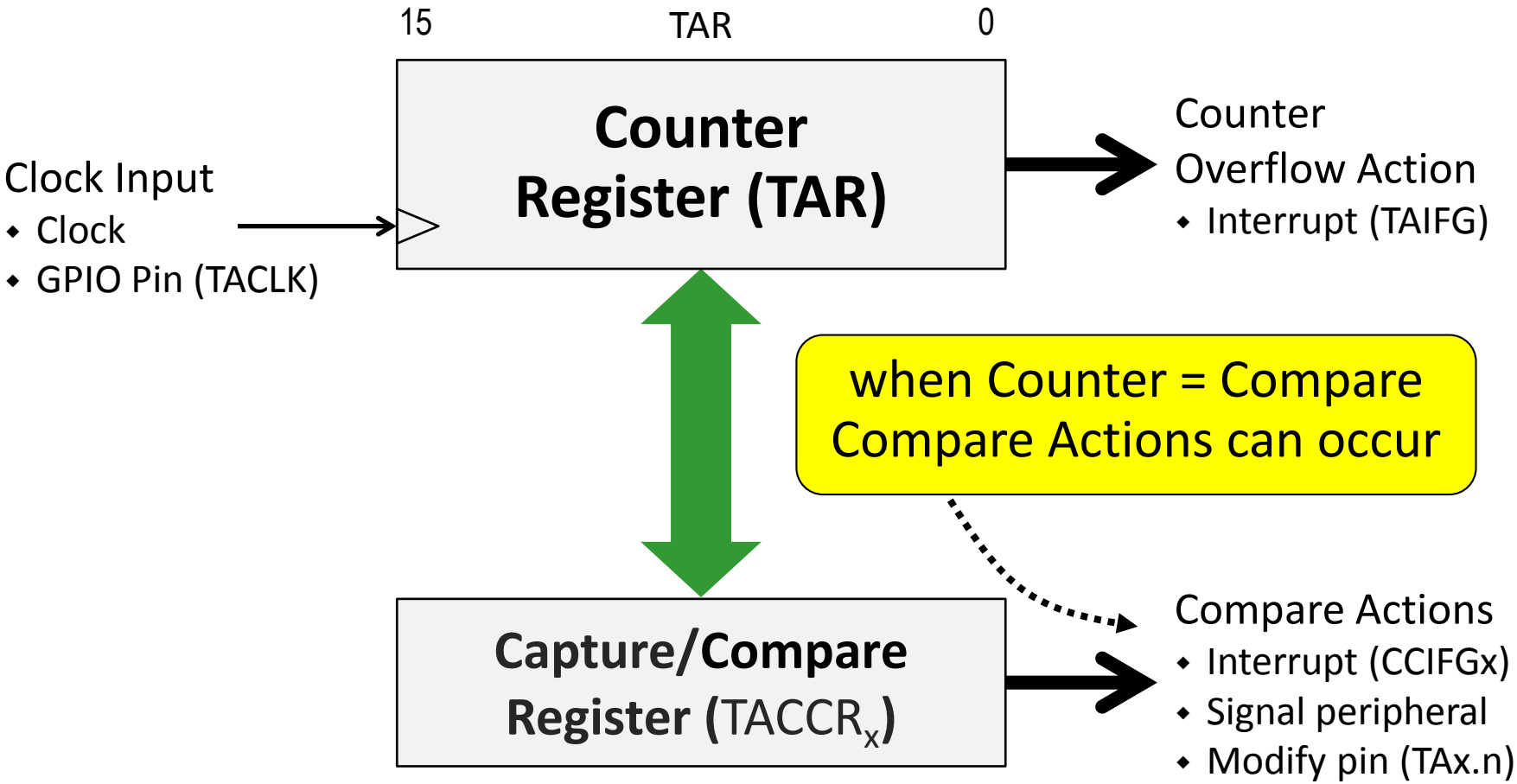
- Compare mode:
  - Compare the value of TAR with the value stored in TACCRx and update an output when they match
- Capture mode: used to record time events
  - Records the “time” (value in TAR) at which the input changes into TACCRx
  - The input, usually CCIxA and CCIxB, can be either external or internal from another peripheral or software, depending on board connections

Capture Basics



- Capture time (i.e. count value) when Capture Input signal occurs
- When capture is triggered, count value is placed in CCR and an interrupt is generated

Compare Basics



- Capture time (i.e. count value) when Capture Input signal occurs
- When capture is triggered, count value is placed in CCR and an interrupt is generated

TACCTLx

TACCTLx, Capture/Compare Control Register

15	14	13	12	11	10	9	8
CMx		**CCISx**		**SCS**	**SCCI**	**Unused**	**CAP**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	r0	rw-(0)
7	6	5	4	3	2	1	0
OUTMODx			**CCIE**	**CCI**	**OUT**	**COV**	**CCIFG**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

CMx Bit 15-14 Capture mode

- 00 No capture
- 01 Capture on rising edge
- 10 Capture on falling edge
- 11 Capture on both rising and falling edges

CCISx Bit 13-12 Capture/compare input select. These bits select the TACCRx input signal. See the device-specific data sheet for specific signal connections.

- 00 CCIxA
- 01 CCIxB
- 10 GND
- 11 V_{cc}

SCS Bit 11 Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock.

- 0 Asynchronous capture
- 1 Synchronous capture

SCCI Bit 10 Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit

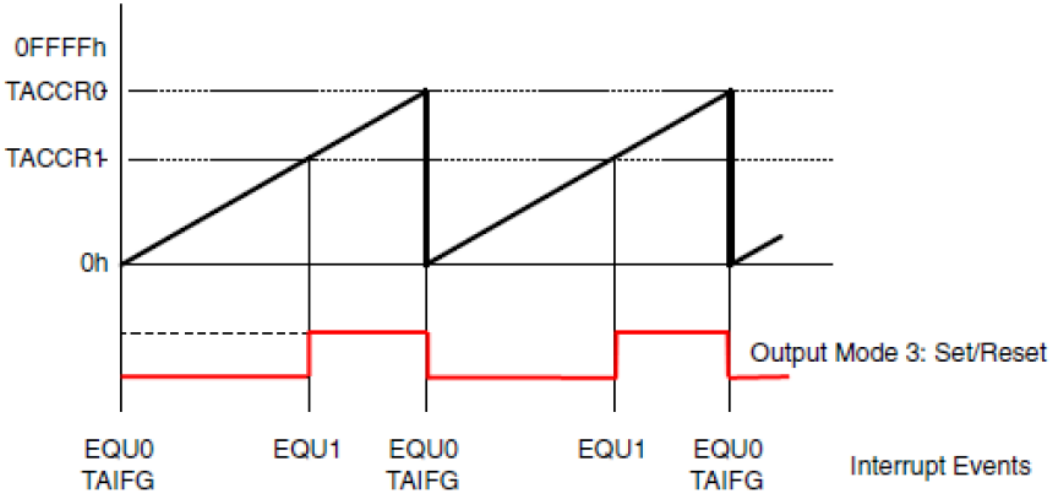
Unused Bit 9 Unused. Read only. Always read as 0.

TACCTL cont'd

CAP	Bit 8	Capture mode
		0 Compare mode
		1 Capture mode
OUTMODx	Bits 7-5	Output mode. Modes 2, 3, 6, and 7 are not useful for TACCR0, because EQUx = EQU0.
		000 OUT bit value
		001 Set
		010 Toggle/reset
		011 Set/reset
		100 Toggle
		101 Reset
		110 Toggle/set
		111 Reset/set
CCIE	Bit 4	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag.
		0 Interrupt disabled
		1 Interrupt enabled
CCI	Bit 3	Capture/compare input. The selected input signal can be read by this bit.
OUT	Bit 2	Output. For output mode 0, this bit directly controls the state of the output.
		0 Output low
		1 Output high
COV	Bit 1	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software.
		0 No capture overflow occurred
		1 Capture overflow occurred
CCIFG	Bit 0	Capture/compare interrupt flag
		0 No interrupt pending
		1 Interrupt pending

Example of Compare Mode

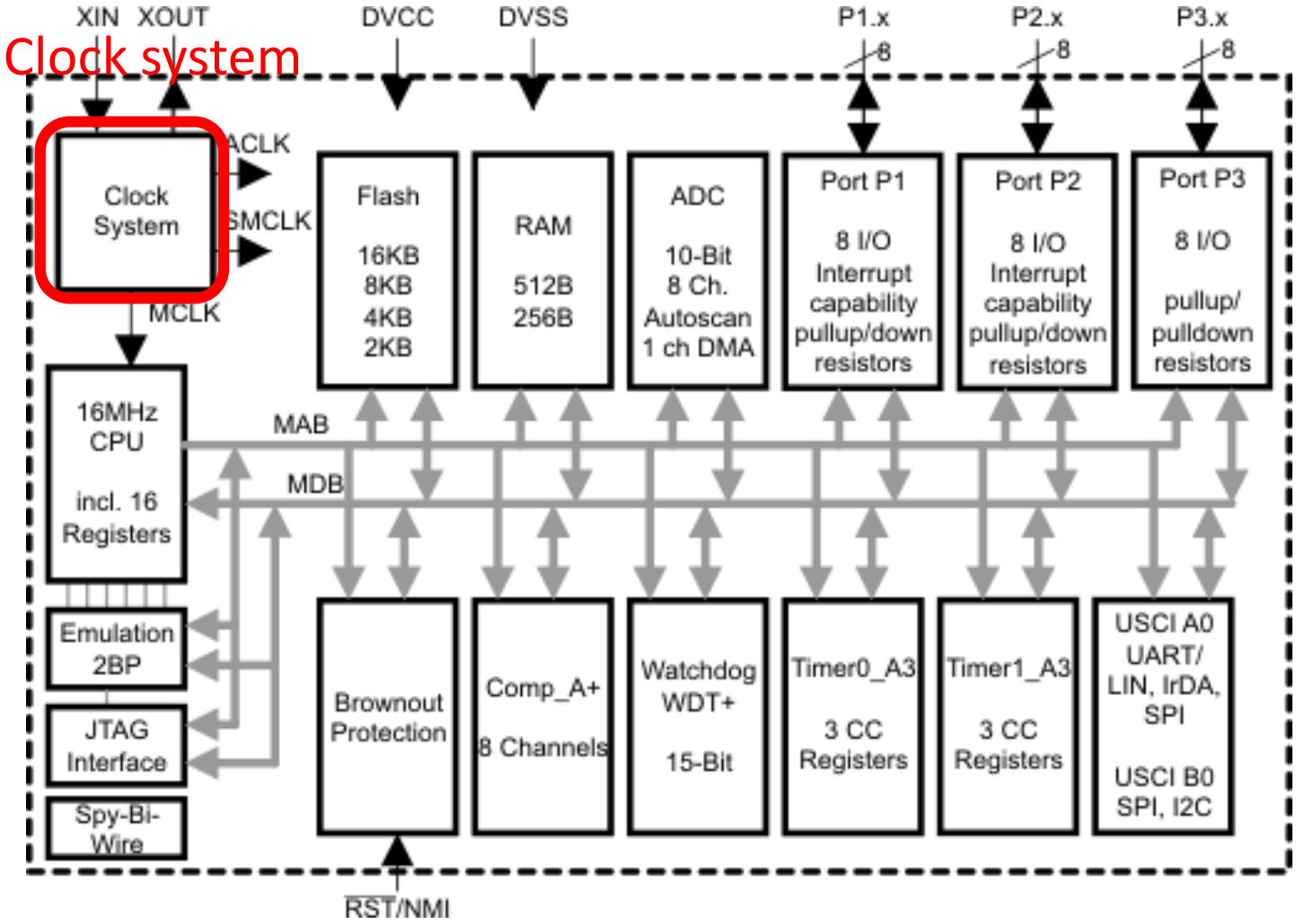
- Exact behavior of a Capture/Compare Block depends on setting of the corresponding control register, e.g.
  - TAR counts to TACCR0 and resets (i.e., TACCR0 determines frequency (along with TAR input frequency))
  - At Output Mode 3, EQU1 is set when TAR>TACCR1 (i.e., TACCR1 determines pulse width)



Lab 3

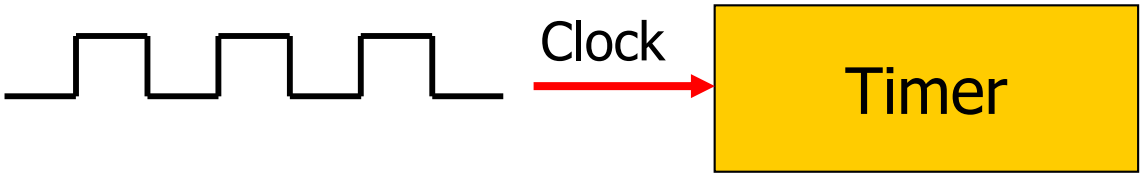
- **Basic 1:**
  - Flash both red and green LEDs at 1 Hz. The green LED should be on for 0.5 sec and off for 0.5 sec. The red LED should be on for 0.2 sec and off for 0.8 sec.
  - Use TAR to keep a cycle time of 1 sec. Use TACCR1 to control the green LED and TACCR2 to control the red LED.
- **Bonus:**
  - Flash the green LED at 1 Hz by polling Timer_A. When the button is pressed, run Basic 1 for 4.8 sec. Then, return to Flash the green LED at 1 Hz.

MSP430 Clock System



Needs for Clocking

- A timer is no more than a counter and has no direct concept of time.
- It is the programmer's job to establish a relation between the value in the counter and real time.
  - This depends on the frequency of the clock for the timer.



- A clock is a square wave signal whose edges trigger hardware
  - Systems usually have conflicting requirements for clocks

Different Requirements for Clocks

- Devices often in a low-power mode until some event occurs, then must wake up and handle event rapidly
  - Clock must get to be stabilized quickly
- Devices also need to keep track of real time: (1) can wake up periodically, or (2) time-stamp external events
- Therefore, two kinds of clocks often needed:
  - A fast clock to drive CPU, which can be started and stopped rapidly but need not be particularly accurate
  - A slow clock that runs continuously to monitor real time, which must use little power and be accurate

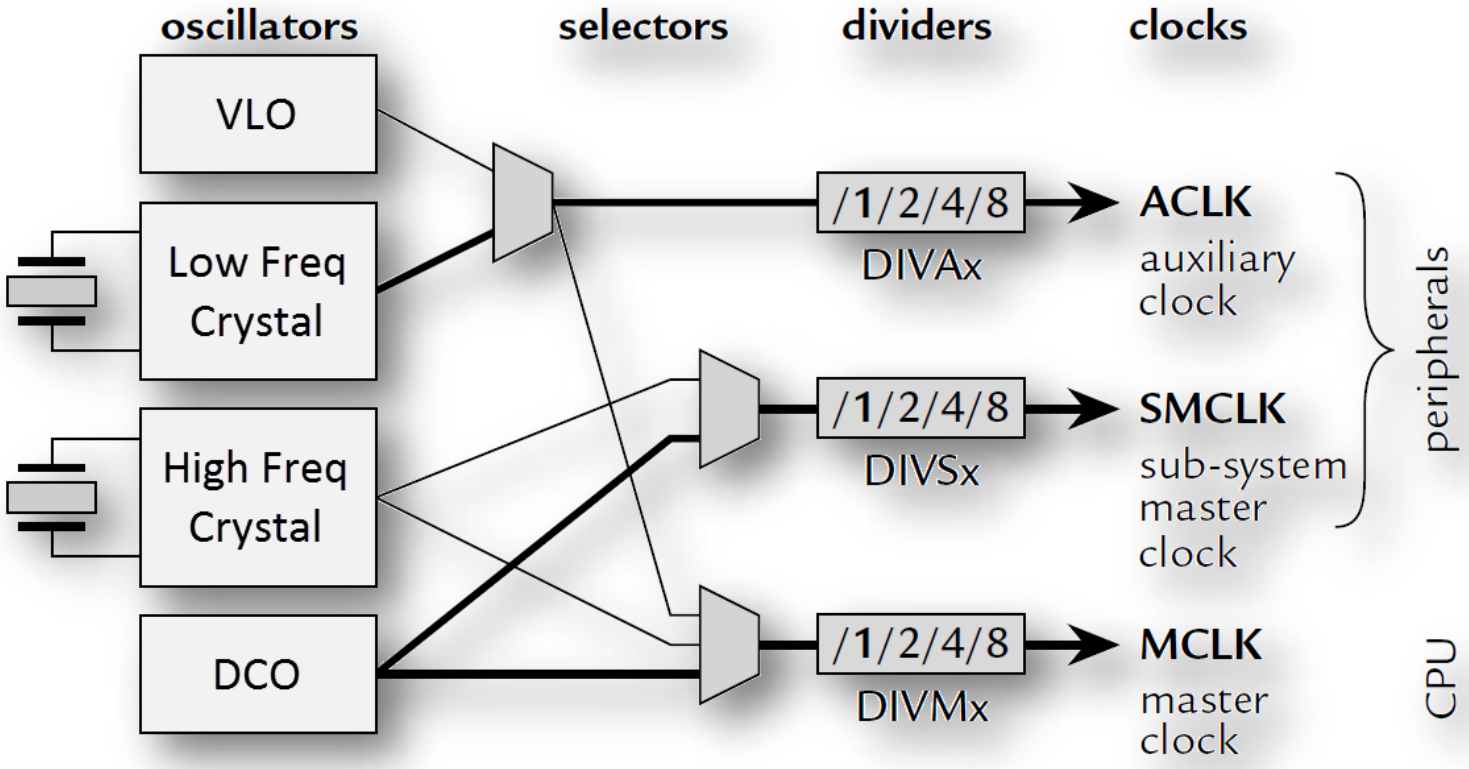
Different Requirements for Clocks

- Different clock sources also have different characteristics
  - *Crystal*: accurate and stable (w.r.t. temperature or time); expensive, delicate, drawing large current, external component, longer time to start up/stabilize
  - *Resistor and capacitor* (RC): cheap, quick to start, integrated within microcontroller and sleep with CPU; poor accuracy and stability
  - *Ceramic resonator and MEMS* clocks in between

Need multiple clocks

Clock System of MSP430

- Variety of osc sources – on-chip (cheap, reliable) and off-chip (accurate)
- Rich selection of oscillator sources routed to internal clocks



Clocks in MSP430

Name	Description	Used-by	Typical Speed
MCLK	Master Clock	CPU	Fast
SMCLK	Sub-Master Clock	Peripherals	Fast
ACLK	Auxiliary Clock	Peripherals	Slow

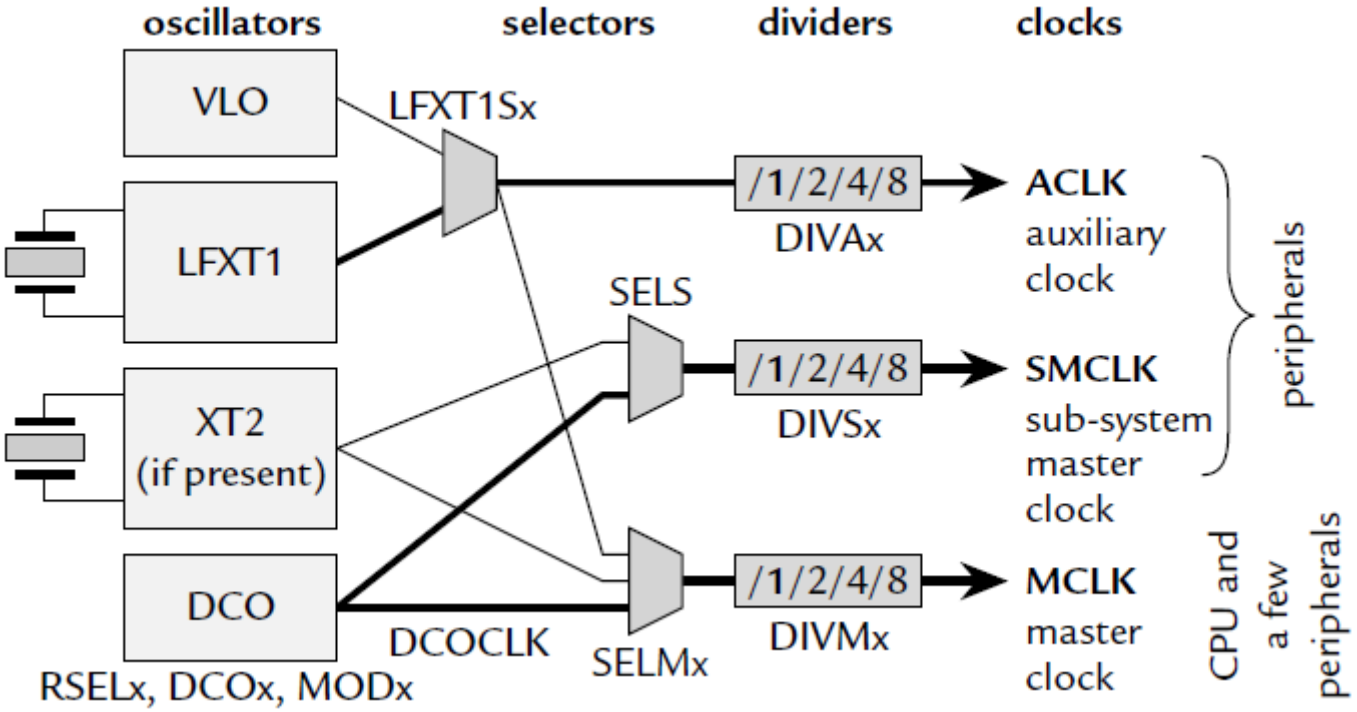
- Master clock (MCLK): for CPU and some peripherals, normally driven by *digitally controlled oscillator* (DCO)
- Subsystem master clock (SMCLK): distributed to peripherals, normally driven by DCO
- Auxiliary clock (ACLK): distributed to peripherals, normally for real-time clocking and driven by a low-frequency crystal oscillator, typically at 32 KHz

Clock Sources

- Low- or high-frequency crystal oscillator, LFXT1:
  - External; used with a low- or high frequency crystal; an external clock signal can also be used; connected to MSP430 through XIN and XOUT pins
- High-frequency crystal oscillator, XT2:
  - External; similar to LFXT1 but at high frequencies
- Very low-power, low-frequency oscillator, VLO:
  - Internal at 12 KHz; alternative to LFXT1 when accuracy of a crystal is not needed; may not available in all devices
- Digitally controlled oscillator, DCO:
  - Internal; a highly controllable RC oscillator that starts fast

From Sources to Clocks

- Typical sources of clocks:
  - MCLK, SMCLK: DCO (typically at 1.1 MHz)
  - ACLK: LFXT 1 (typically at 32 KHz)



Controlling Clocks

- In MSP430, the Basic Clock Module is also an IO peripheral and can be controlled by registers, DCOCTL and BCSCTL1–3
  - DCOCTL (056h): configure DCO
  - BCSCTL1 (basic clock system control 1, 057h): configure ACLK
  - BCSCTL2 (basic clock system control 2, 058h): configure MCLK, SMCLK
  - BCSCTL3 (basic clock system control 3, 053h): control LFXT1/VLO

Control Registers for Clocks

Control Registers for Clock System

Table 5-1. Basic Clock Module+ Registers

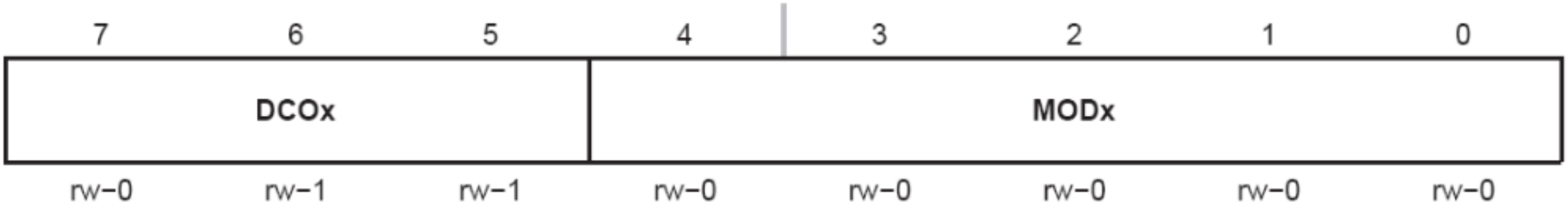
Register	Short Form	Register Type	Address	Initial State
DCO control register	DCOCTL	Read/write	056h	060h with PUC
Basic clock system control 1	BCSCTL1	Read/write	057h	087h with POR(1)
Basic clock system control 2	BCSCTL2	Read/write	058h	Reset with PUC
Basic clock system control 3	BCSCTL3	Read/write	053h	005h with PUC(2)
SFR interrupt enable register 1	IE1	Read/write	000h	Reset with PUC
SFR interrupt flag register 1	IFG1	Read/write	002h	Reset with PUC

(1) Some of the register bits are also PUC initialized (see Section 5.3.2).
(2) The initial state of BCSCTL3 is 000h in the MSP430AFE2xx devices.

- DCOCTL and BCSCTL1 combined define the frequency of DCO, among other settings

DCOCTL (at Memory Address 056h)

DCOCTL, DCO Control Register



DCOx	Bits 7-5	DCO frequency select. These bits select which of the eight discrete DCO frequencies of the RSELx setting is selected.
MODx	Bits 4-0	Modulator selection. These bits define how often the f_{DCO+1} frequency is used within a period of 32 DCOCLK cycles. During the remaining clock cycles (32−M used. Not useable when DCOx=7.

Tag-Length-Value

```
DCOCTL = CALDCO_1MHZ;          // Set DCO step + modulation
```

Tag-Length-Value

- Tag-Length-Value (TLV) stores device-specific information in the flash memory to set DCOCTL and BCSCTL1 for DCO frequency

DCO Calibration Data (Device Specific)

Label	Description	Offset
CALBC1_1MHZ	Value for the BCSCTL1 register for 1 MHz, T_A = 25°C	0x07
CALDCO_1MHZ	Value for the DCOCTL register for 1 MHz, T_A = 25°C	0x06
CALBC1_8MHZ	Value for the BCSCTL1 register for 8 MHz, T_A = 25°C	0x05
CALDCO_8MHZ	Value for the DCOCTL register for 8 MHz, T_A = 25°C	0x04
CALBC1_12MHZ	Value for the BCSCTL1 register for 12 MHz, T_A = 25°C	0x03
CALDCO_12MHZ	Value for the DCOCTL register for 12 MHz, T_A = 25°C	0x02
CALBC1_16MHZ	Value for the BCSCTL1 register for 16 MHz, T_A = 25°C	0x01
CALDCO_16MHZ	Value for the DCOCTL register for 16 MHz, T_A = 25°C	0x00

```
BCSCTL1 = CALBC1_1MHZ;                    // Set range
DCOCTL = CALDCO_1MHZ;
```

BCSCTL1

BCSCTL1, Basic Clock System Control Register 1

7	6	5	4	3	2	1	0
XT2OFF	**XTS**(1)(2)	**DIVAx**		**RSELx**			
rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-1	rw-1	rw-1

XT2OFF	Bit 7	XT2 off. This bit turns off the XT2 oscillator 0 XT2 is on 1 XT2 is off if it is not used for MCLK or SMCLK.
XTS	Bit 6	LFXT1 mode select. 0 Low-frequency mode 1 High-frequency mode
DIVAx	Bits 5-4	Divider for ACLK 00 /1 01 /2 10 /4 11 /8
RSELx	Bits 3-0	Range select. Sixteen different frequency ranges are available. The lowest frequency range is selected by setting RSELx=0. RSEL3 is ignored when DCOR = 1.

```
BCSCTL1 = CALBC1_1MHZ;                    // Set range
```

BCSCTL2

BCSCTL2, Basic Clock System Control Register 2

MCLK **SMCLK**

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		DCOR(1)(2)
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

SELMx	Bits 7-6	Select MCLK. These bits select the MCLK source.	
		00	DCOCLK
		01	DCOCLK
		10	XT2CLK when XT2 oscillator present on-chip. LFXT1CLK or VLOCLK when XT2 oscillator not present on-chip.
		11	LFXT1CLK or VLOCLK
DIVMx	Bits 5-4	Divider for MCLK	
		00	/1
		01	/2
		10	/4
		11	/8
SELS	Bit 3	Select SMCLK. This bit selects the SMCLK source.	
		0	DCOCLK
		1	XT2CLK when XT2 oscillator present. LFXT1CLK or VLOCLK when XT2 oscillator not present
DIVSx	Bits 2-1	Divider for SMCLK	
		00	/1
		11	/8

```
BCSCTL2 |= SELM_3 + DIVM_3;          // MCLK = VLO/8
```

BCSCTL3

BCSCTL3, Basic Clock System Control Register 3

In MSP430G2231

7	6	5	4	3	2	1	0
XT2Sx		LFXT1Sx		XCAPx[1]		XT2OF[2]	LFXT1OF[1]
rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	r0	r-(1)

XT2Sx Bits 7-6 XT2 range select. These bits select the frequency range for XT2.

00 0.4- to 1-MHz crystal or resonator
01 1- to 3-MHz crystal or resonator
10 3- to 16-MHz crystal or resonator
11 Digital external 0.4- to 16-MHz clock source

LFXT1Sx Bits 5-4 Low-frequency clock select and LFXT1 range select. These bits select between LFXT1 and VLO when XTS = 0, and select the frequency range for LFXT1 when XTS = 1.

When XTS = 0:
00 32768-Hz crystal on LFXT1
01 Reserved
10 VLOCLK (Reserved in MSP430x21x1 devices)
11 Digital external clock source

When XTS = 1 (Not applicable for MSP430x20xx devices)
00 0.4- to 1-MHz crystal or resonator
01 1- to 3-MHz crystal or resonator
10 3- to 16-MHz crystal or resonator

```
BCSCTL3 |= LFXT1S_2;     // Enable VLO as MCLK/ACLK src
```

01 Reserved

Interrupt Flag Register 1 (IFG1)

- OFIFG oscillator-fault flag is set when an oscillator fault (LFXT1OF) is detected.

IFG1, Interrupt Flag Register 1

7	6	5	4	3	2	1	0
						OFIFG	
						rw-1	

	Bits 7-2	These bits may be used by other modules. See device-specific data sheet.
OFIFG	Bit 1	Oscillator fault interrupt flag. Because other bits in IFG1 may be used for other modules, it is recommended to set or clear this bit using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions.
		0 No interrupt pending
		1 Interrupt pending
	Bits 0	This bit may be used by other modules. See device-specific data sheet.

```
IFG1 &= ~OFIFG;                    // Clear OSCFault flag
```

Recall Sample Code for Timer_A

- Flash red LED at 1 Hz if SMCLK at 800 KHz

```
#include <msp430g2553.h>
#define LED1 BIT0
void main (void) {
  WDTCTL = WDTPW|WDTHOLD; // Stop watchdog timer
  P1OUT = ~LED1;  P1DIR = LED1;  TACCR0 = 49999;
  TACTL = MC_1|ID_3|TASSEL_2|TACLR; //Setup Timer_A
  //up mode, divide clk by 8, use SMCLK, clr timer
  for (;;) { // Loop forever
    while (!(TACTL&TAIFG)) { // Wait time up
    } // doing nothing
    TACTL &= ~TAIFG;  // Clear overflow flag
    P1OUT ^= LED1;    // Toggle LEDs
  } // Back around infinite loop
}
```

Sample Code for Setting Clocks

- Set DCO to 1MHz, enable crystal

```
#include <msp430g2231.h> (#include <msp430g2553.h> )
void main(void) {
  WDTCTL = WDTPW + WDTHOLD;  // Stop watchdog timer
  if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
     while(1);   // If TLV erased, TRAP!
  BCSCTL1 = CALBC1_1MHZ;     // Set range
  DCOCTL = CALDCO_1MHZ;
  P1DIR = 0x41;  // P1.0 & 6 outputs (red/green LEDs)
  P1OUT = 0x01;  // red LED on
  BCSCTL3 |= LFXT1S_0; // Enable 32768 crystal
  IFG1 &= ~OFIFG;// Clear OSCFault flag
  P1OUT = 0;     // red LED off
  BCSCTL2 |= SELS + DIVS_3; // SMCLK = DCO/8
  // infinite loop to flash LEDs
}
```

Lab 3

- **Basic 2:**
  - Flash the green LED at 1 Hz by polling Timer_A, which is driven by ACLK sourced by VLO.
  - Hint: Since TAR register is 16-bit (0~65535) long, you should be careful of its overflow by using clock source “Divider”.