

# Controlling Timer Interrupt Frequencies



Exploring the Basic Clock Module  
system in order to determine how to  
control timer frequencies

# Goal of this Lecture

---

- This supports Lab 2: Flashing LED using interrupts.
- You have seen existing code which performs this. However, you don't know the exact frequency of the flashing LED.
- Recall the existing code.....  
....a big problem is that this code must go to low power mode before the LED flashes.

# Sample LED Flash Code

```
#include <msp430x20x3.h>
```

```
void main(void)
```

```
{
```

```
    WDCTL = WDTPW + WDTHOLD;
```

```
    P1DIR |= 0x01;
```

```
    CCTLO = CCIE;
```

```
    CCR0 = 50000;
```

```
    TACTL = TASSEL_2 + MC_2;
```

```
    _BIS_SR(LPM0_bits + GIE);
```

```
}
```

```
// Timer A0 interrupt service routine
```

```
#pragma vector=TIMER_A0_VECTOR
```

```
__interrupt void Timer_A (void)
```

```
{
```

```
    P1OUT ^= 0x01;
```

```
    CCR0 += 50000;
```

```
}
```

Should be encapsulated in an initialization routine....like "TIMER\_init()"

```
// Stop WDT
```

```
// P1.0 output
```

```
// CCR0 interrupt enabled
```

```
// SMCLK, contmode
```

```
// Enter LPM0 w/ interrupt
```

This is a problem – you do NOT want to go to low power mode, but you DO need interrupts turned on...see end of TIMER lecture.

```
// Toggle P1.0
```

```
// Add Offset to CCR0
```

No hard coded values. Use variables to control the interrupt firing rate.

**THE BIG QUESTION:**  
What time interval (frequency) does 50,000 correspond to???

# How Fast is 50,000 counts?

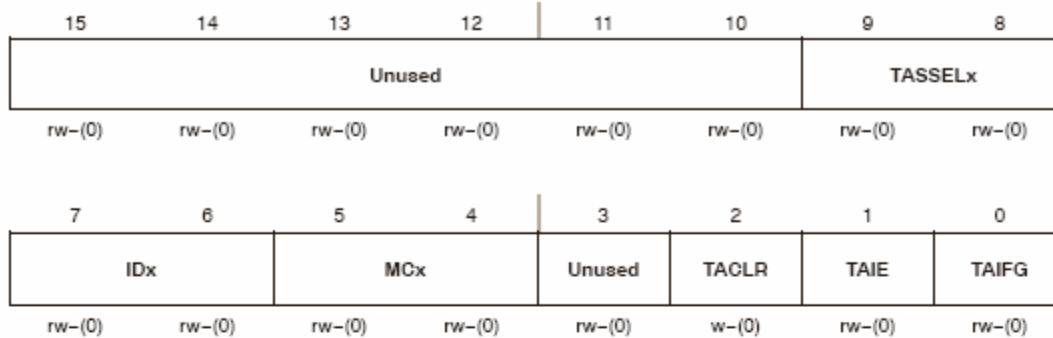
- To appropriately control interrupt frequencies, need to understand how the timer reload value relates to actual time.
- Question: 1 tick (or one count of reload TACCR0) = ??? seconds
- To understand this, need to understand the clock sources and clock signals in the system

# TACTL: Timer Control Register

---

- From last lecture, recall that the timer must be configured.
- We use TASSSELx to tell the processor which internal clock signal we want to use to drive the TIMER.
- Default is the SMCLK – Subsystem Master Clock
- See next slide to show how this is configured ...

## TACTL, Timer\_A Control Register



Unused Bits 15-10

TASSELx	Bits 9-8	Timer_A clock source select
	00	TACLK
	01	ACLK
	10	SMCLK
	11	INCLK

We must tell it we are using the Subsystem Clock, SMCLK. This is #define TASSEL\_2 in header file.

IDx Bits 7-6 Input divider. These bits select the divider for the input clock.

00	/1
01	/2
10	/4
11	/8

MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power.
	00	Stop mode: the timer is halted
	01	Up mode: the timer counts up to TACCR0
	10	Continuous mode: the timer counts up to 0FFFFh
	11	Up/down mode: the timer counts up to TACCR0 then down to 0000h

Select Continuous Mode, 2.  
#define MC\_2

Unused Bit 3 Unused

TACLr Bit 2 Timer\_A clear. Setting this bit resets TAR, the clock divider, and the count direction. The TACLr bit is automatically reset and is always read as zero.

TAIE Bit 1 Timer\_A interrupt enable. This bit enables the TAIFG interrupt request.

0	Interrupt disabled
1	Interrupt enabled

TAIFG Bit 0 Timer\_A interrupt flag

0	No interrupt pending
1	Interrupt pending

# What is SMCLK and Where does it Come From?

---

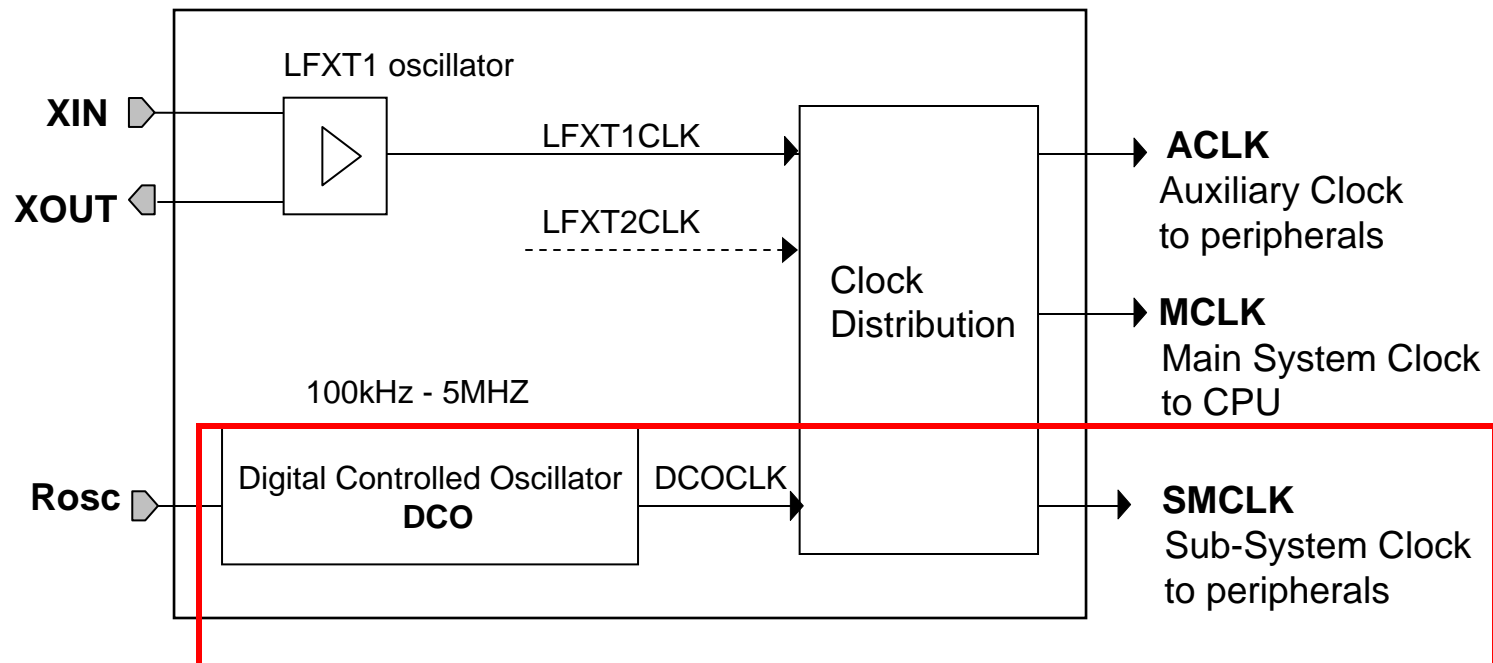
- So now that we know SMCLK controls the timer frequency, how do we know what SMCLK frequency is?
- Have to understand the basic clock subsystem.
- Processor runs on external clock sources, and produces internal clock signals. Everything runs off the internal clock signals: ACLK, MCLK, and SMCLK.

# Basic Clock Module

- Clock Sources
  - One DCO, internal digitally controlled oscillator (aka DCOCLK)
    - Generated on-chip RC-type, frequency controlled by SW + HW
    - Freq ~ 1.1MHz
  - One LF/XT oscillator
    - LF: 32768Hz
- Clocks Signals Available from Clock Module:
  - ACLK auxiliary clock ACLK
  - MCLK main system clock MCLK
  - SMCLK sub main system clock – **USED FOR TIMERS**
- Why so many clock signals?
  - So user can choose higher frequency for fast response, and use lower frequency to allow processor to go to sleep and/or conserve power.

**These clock SOURCES are used to create these clock SIGNALS**

# Clocks and Signals

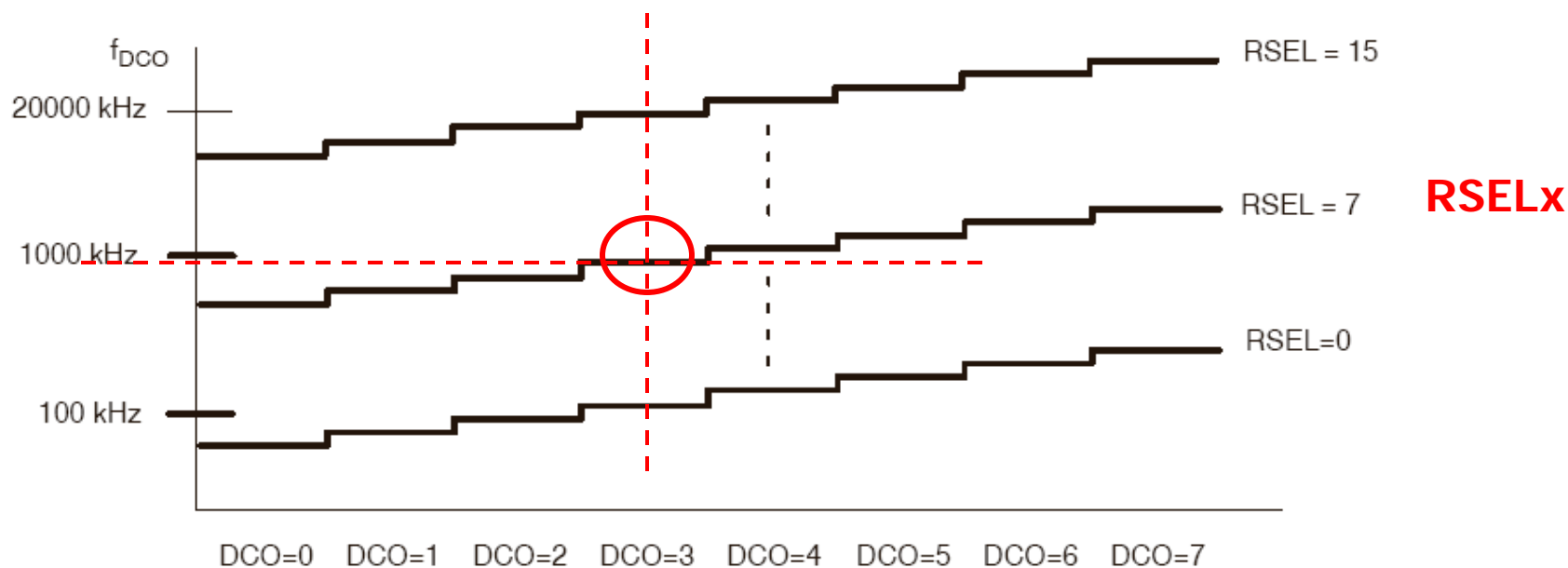


See User Guide Figure 4-1 Basic Clock Module+ Block Diagram (next page)

**SMCLK sources the TIMER interrupt. It gets its frequency from DCOCLK. So first, find out frequency of DCOCLK.**

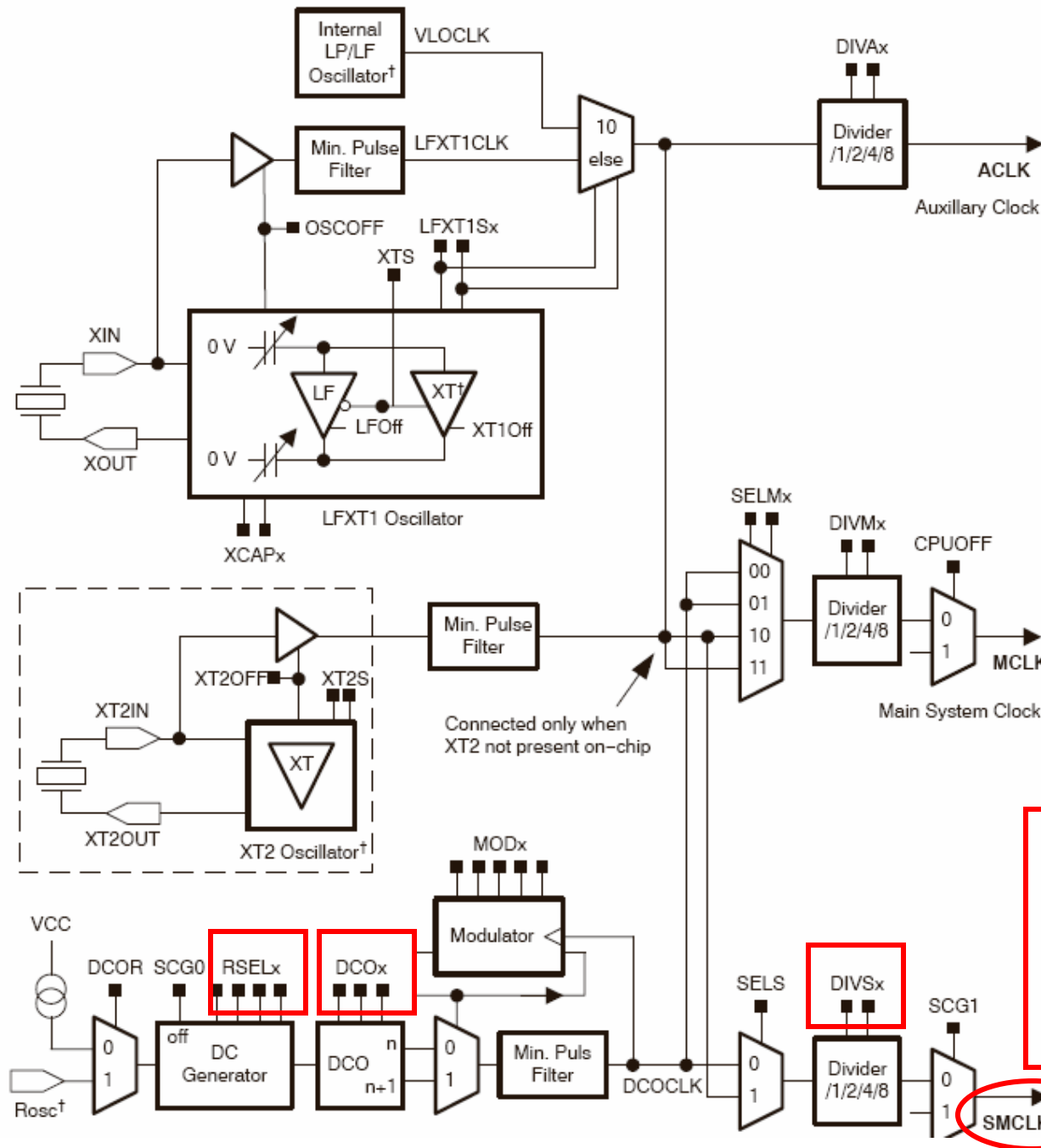
# DCO – Digitally Controlled Oscillator

- DCOCLK frequency controlled in SW by DCOx, MODx, and RSELx bits
- Lots of fine tuning of the frequency is possible
- Default values: RSELx = 7, DCOx = 3 (we must confirm these)
- DCO default frequency is 1 ~MHz



**DCOx**

Figure 4-1. Basic Clock Module+ Block Diagram



This complicated figure shows how the clock signals (ACLK, MCLK, SMCLK) are configured. Many parameters allow fine tuning.

Three critical parameters for SMCLK: RSELx, DCOx, and DIVSx. We need to figure out what these are.

# Registers to Configure Clocks

These are the registers that configure the clocks. We will look at each one of them individually to see where the critical parameters are located.

*Table 4–1. Basic Clock module+ Registers*

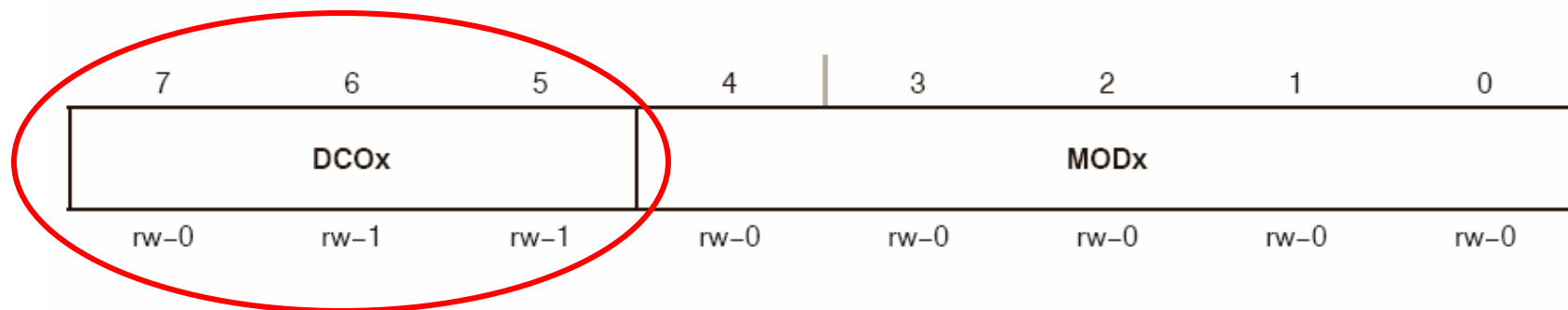
Register	Short Form	Register Type	Address	Initial State
DCO control register	DCOCTL	Read/write	056h	060h with PUC
Basic clock system control 1	BCSCTL1	Read/write	057h	087h with POR
Basic clock system control 2	BCSCTL2	Read/write	058h	Reset with PUC
Basic clock system control 3	BCSCTL3	Read/write	053h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

We can find the values of each register from the debugger to see how the clock subsystem is configured, so we can verify the TIMER timing.

**What are RSELx, DCOx, and DIVSx?**

# DCOCTL: Control Register for DCO

**DCOCTL, DCO Control Register**



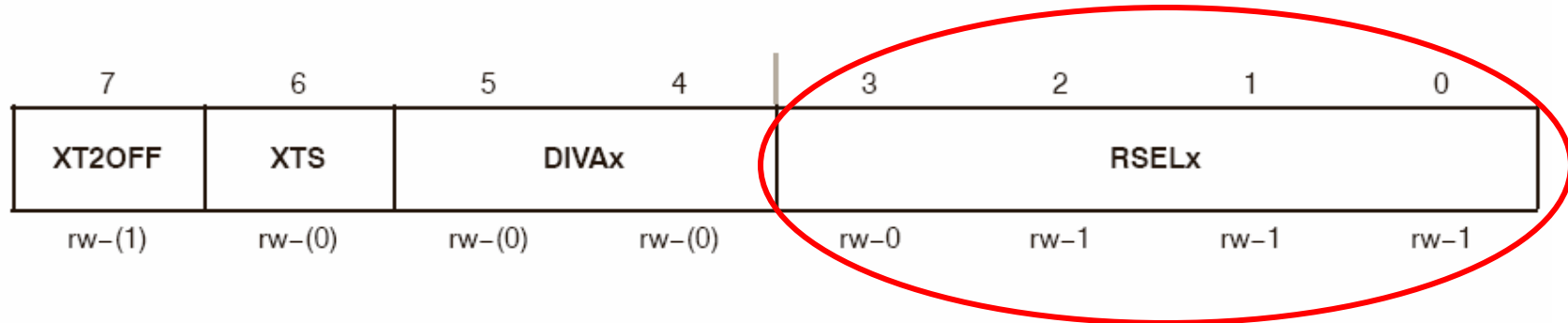
<b>DCOx</b>	Bits 7-5	DCO frequency select. These bits select which of the eight discrete DCO frequencies within the range defined by the RSELx setting is selected.
<b>MODx</b>	Bits 4-0	Modulator selection. These bits define how often the $f_{DCO+1}$ frequency is used within a period of 32 DCOCLK cycles.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

From debugger, DCOCTL = 0x60 = 0110 0000

Therefore, DCOx = 011 = 3

# BCSCTL1: Clock Control Reg 1

## BCSCTL1, Basic Clock System Control Register 1



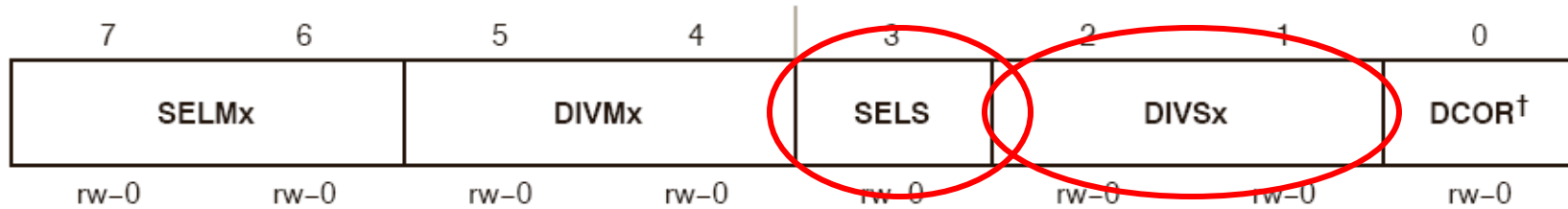
XT2OFF	Bit 7	XT2 off. This bit turns off the XT2 oscillator 0 XT2 is on 1 XT2 is off if it is not used for MCLK or SMCLK.
XTS	Bit 6	LFXT1 mode select. 0 Low frequency mode 1 High frequency mode
DIVAx	Bits 5-4	Divider for ACLK 00 /1 01 /2 10 /4 11 /8

BCSCTL1 = 0x87 = 1000 0111  
Therefore, RSELx = 0111 = 7

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.
-------	----------	--

# BCSCTL2: Clock Control Reg 2

## BCSCTL2, Basic Clock System Control Register 2



† Does not apply to MSP430x20xx or MSP430x21xx.

BCSCTL2 = 0x00 = 0000 0000

Therefore, SELS = 0 and DIVs = 0

<b>SELMx</b>	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	<b>DIVSx</b>	Bits 2-1
<b>DIVMx</b>	Bits 5-4	Divider for MCLK		Divider for SMCLK
		00 /1		00 /1
		01 /2		01 /2
		10 /4		10 /4
		11 /8		11 /8
			<b>DCOR</b>	Bit 0
				DCO resistor select
				0 Internal resistor
				1 External resistor
<b>SELS</b>	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		

# So what did we learn?

- DCOCLK is configured at approximately 1 MHz  
(**RELSs = 7** and **DCOx = 3**)
- SMCLK is indeed controlled by DCOCLK (**SELS = 0**)
- SMCLK runs at same frequency as DCLCLK (divider = 1)  
(**DIVSx = 0**)
- From User Guide 8.2, we know the timer/counter increments or decrements on the **rising edge of the clock signal**.
- If the clock signal is 1 MHz, rising edges are 1 MHz, which corresponds to a time period of 1 usec.
  
- Therefore, **1 count (aka 1 tick) = 1 usec**
- That means if you use a reload value  $TACCR0 = 50,000$ , then the timer will fire every  
 $50,000 \text{ ticks} * (1 \text{ usec/tick}) = 50,000 \text{ usec} = .05 \text{ seconds}$ .