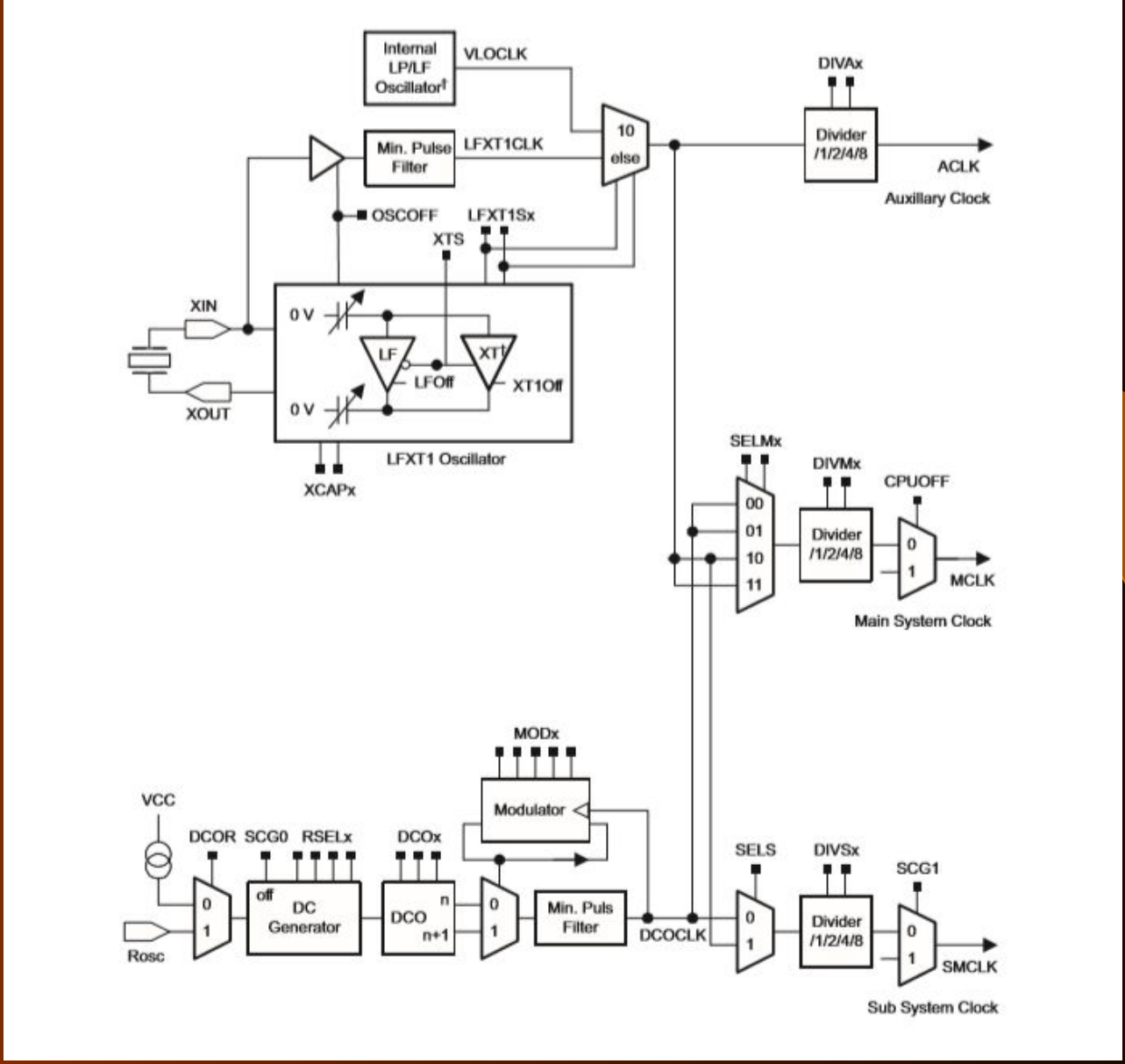
Introduction to Embedded System Design

MSP430 Clock System and Reset

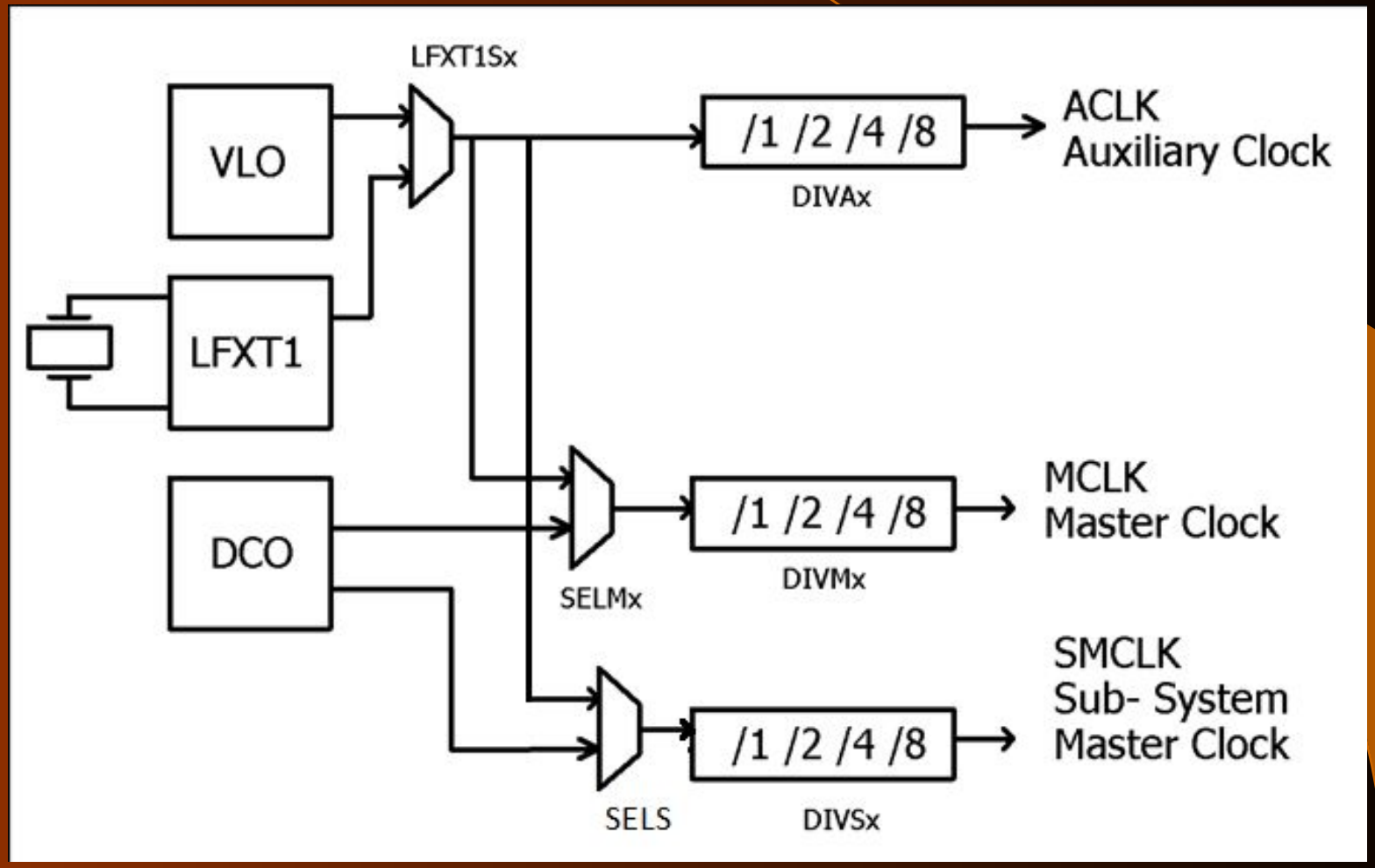
Dhananjay V. Gadre
Associate Professor
ECE Division
Netaji Subhas University of Technology, New Delhi

Badri Subudhi
Assistant Professor
Electrical Engineering Department
Indian Institute of Technology, Jammu

Basic Clock Module



Simplified Block Diagram of MSP430G2553 Clock System



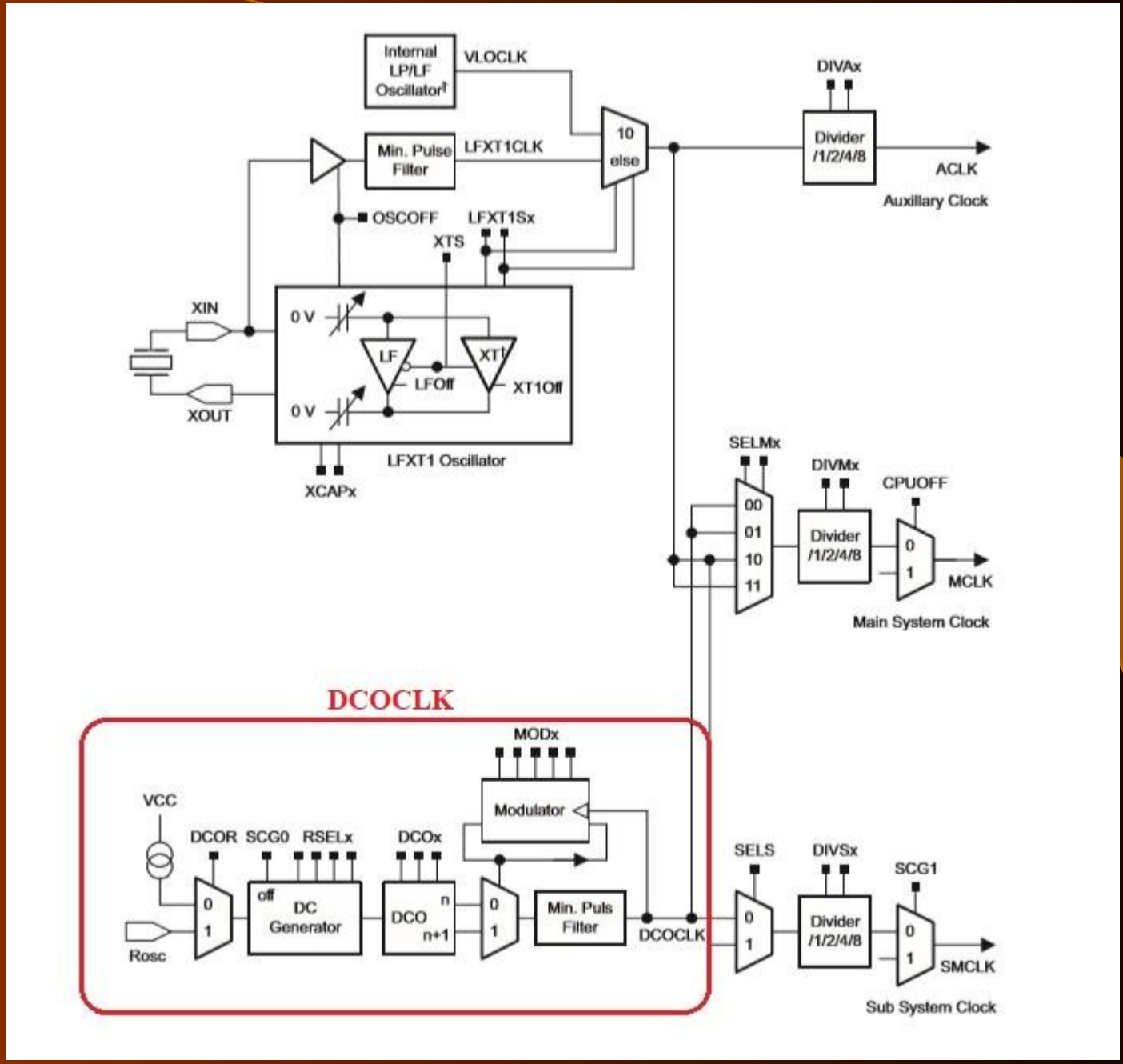
Clock Sources

- **MSP430 can get clock from 3 sources:**
  1. **DCO**
  2. **LFXT1**
  3. **VLO**

Clock Sources

1. **DCO**
   Digitally Controlled Oscillator

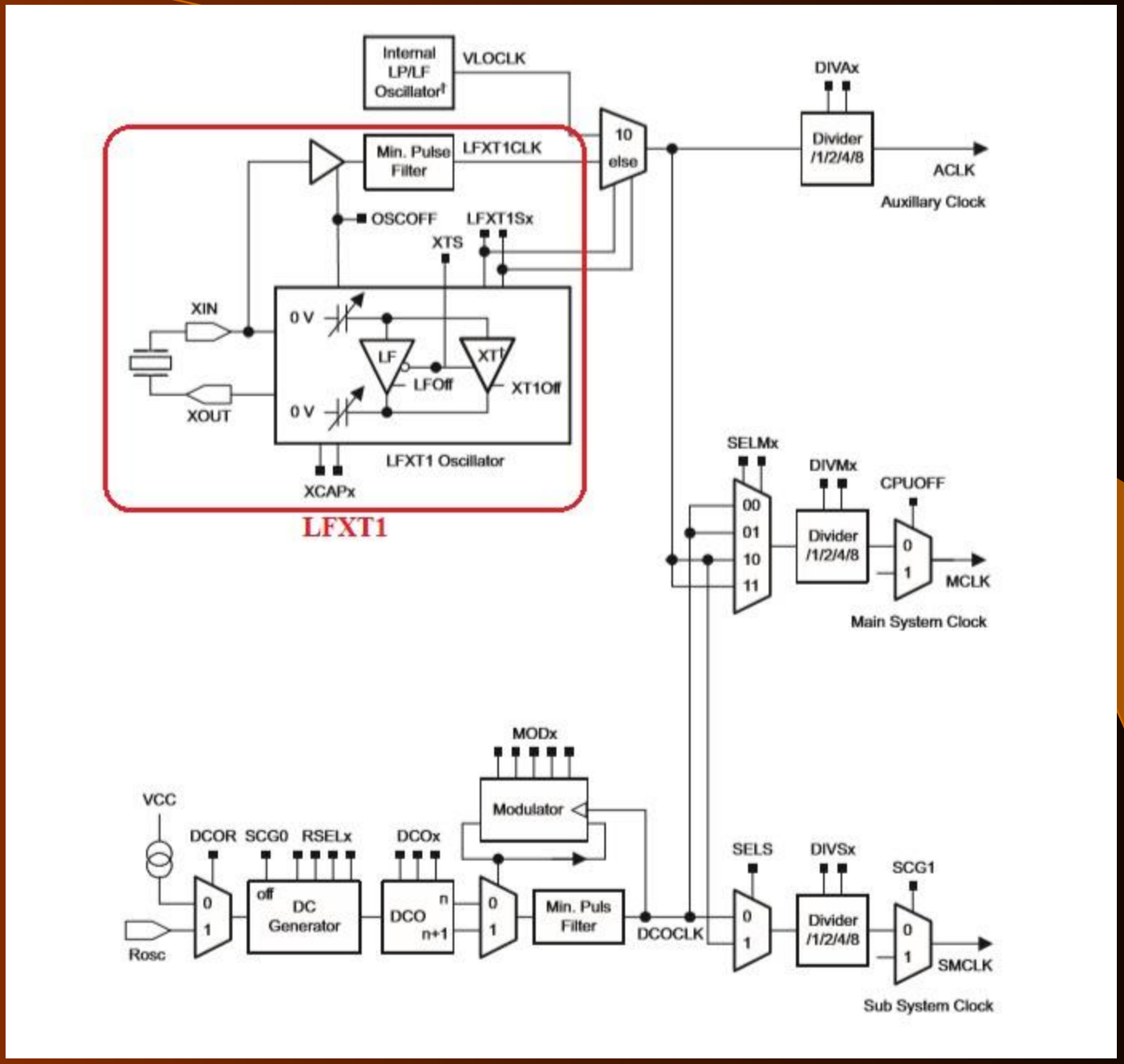
- One of the highlights of MSP430, it is basically a highly controllable RC oscillator that starts in less than 1μs. Therefore starts rapidly at full speed from LPM.
- The DCO frequency can be adjusted by software.
- DCO frequency ranges from 60K to 16MHz.
- **Default frequency => 1.1MHz**


Internal LP/LF Oscillator
VLOCLK
Min. Pulse Filter
LFXT1CLK
10
else
DIVAx
Divider /1/2/4/8
ACLK
Auxiliary Clock
OSCOFF
LFXT1Sx
XTS
XIN
XOUT
0 V
0 V
LF
XT1
LFOff
XT1Off
LFXT1 Oscillator
XCAPx
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
DCOCLK
MODx
Modulator
VCC
DCOR
SCG0
RSELx
DCOx
off
DC Generator
DCO
n
n+1
Rosc
Min. Puls Filter
DCOCLK
SELS
DIVSx
SCG1
Divider /1/2/4/8
SMCLK
Sub System Clock

Clock Sources

2. LFXT1

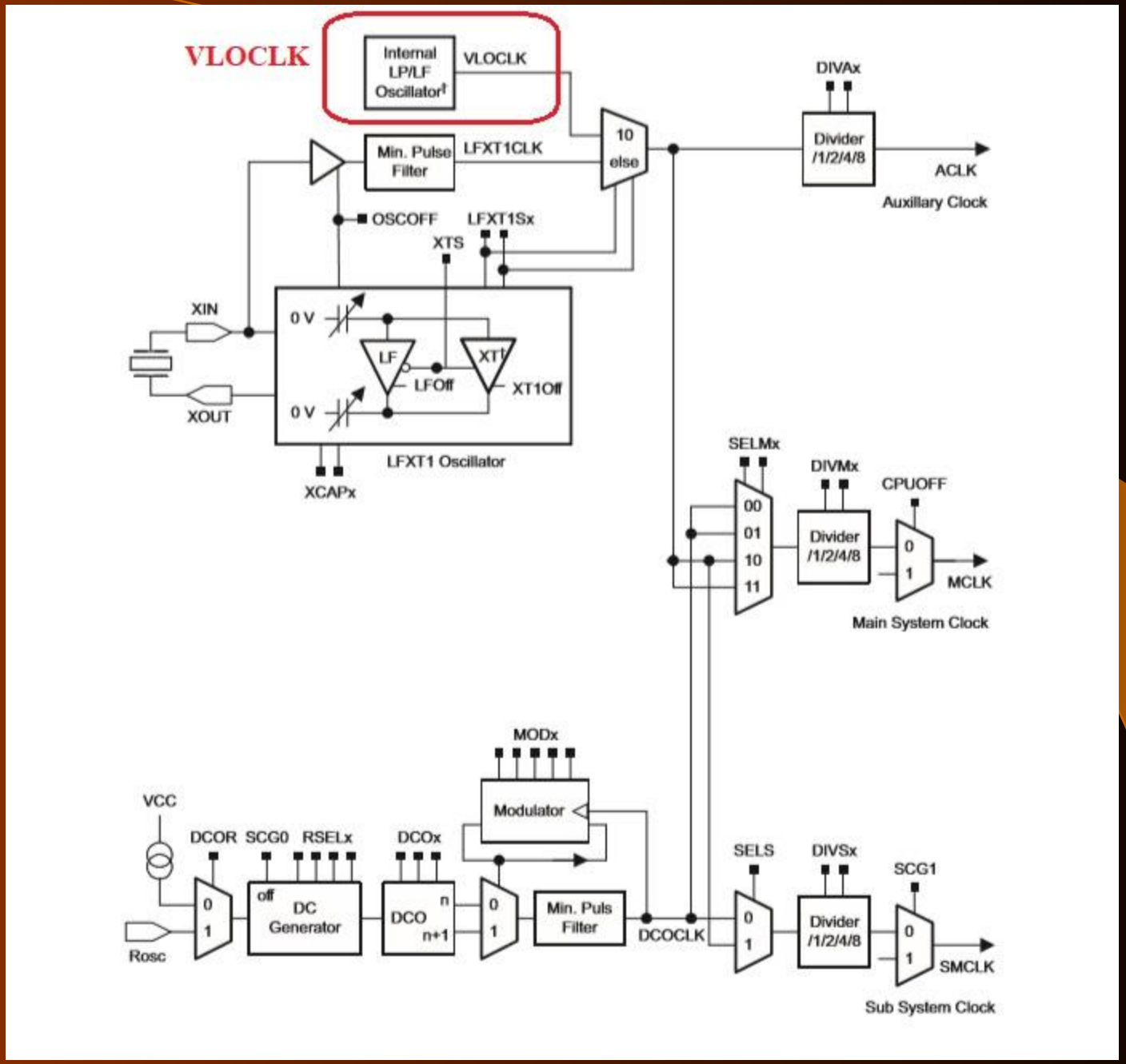
- Low or high frequency crystal oscillator.
- Used with low frequency watch crystal (32 kHz)
- Also used with a high frequency crystal (400 kHz to 16MHz) (Absent in MSP430G2553)


Internal LP/LF Oscillator†
VLOCLK
DIVAx
Min. Pulse Filter
LFXT1CLK
10
else
Divider /1/2/4/8
ACLK
Auxiliary Clock
OSCOFF
LFXT1Sx
XTS
XIN
0 V
LF
XT1
LFOff
XT1Off
XOUT
0 V
LFXT1 Oscillator
XCAPx
LFXT1
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
MODx
Modulator
VCC
DCOR
SCG0
RSELx
DCOx
SELS
DIVSx
SCG1
off
DC Generator
DCO
n
n+1
Min. Puls Filter
DCOCLK
Divider /1/2/4/8
Rosc
SMCLK
Sub System Clock

Clock Sources

3. VLO

- Internal very low-power, low-frequency oscillator
- Typical frequency 12kHz


VLOCLK
Internal LP/LF Oscillator†
VLOCLK
DIVAx
Divider /1/2/4/8
ACLK
Auxiliary Clock
Min. Pulse Filter
LFXT1CLK
10
else
OSCOFF
LFXT1Sx
XTS
XIN
XOUT
0 V
0 V
LF
XT1
LFOff
XT1Off
LFXT1 Oscillator
XCAPx
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
MODx
Modulator
VCC
DCOR
SCG0
RSELx
DCOx
off
DC Generator
DCO
n
n+1
Min. Puls Filter
DCOCLK
SELS
DIVSx
SCG1
Divider /1/2/4/8
SMCLK
Sub System Clock
Rosc

Clock Signals

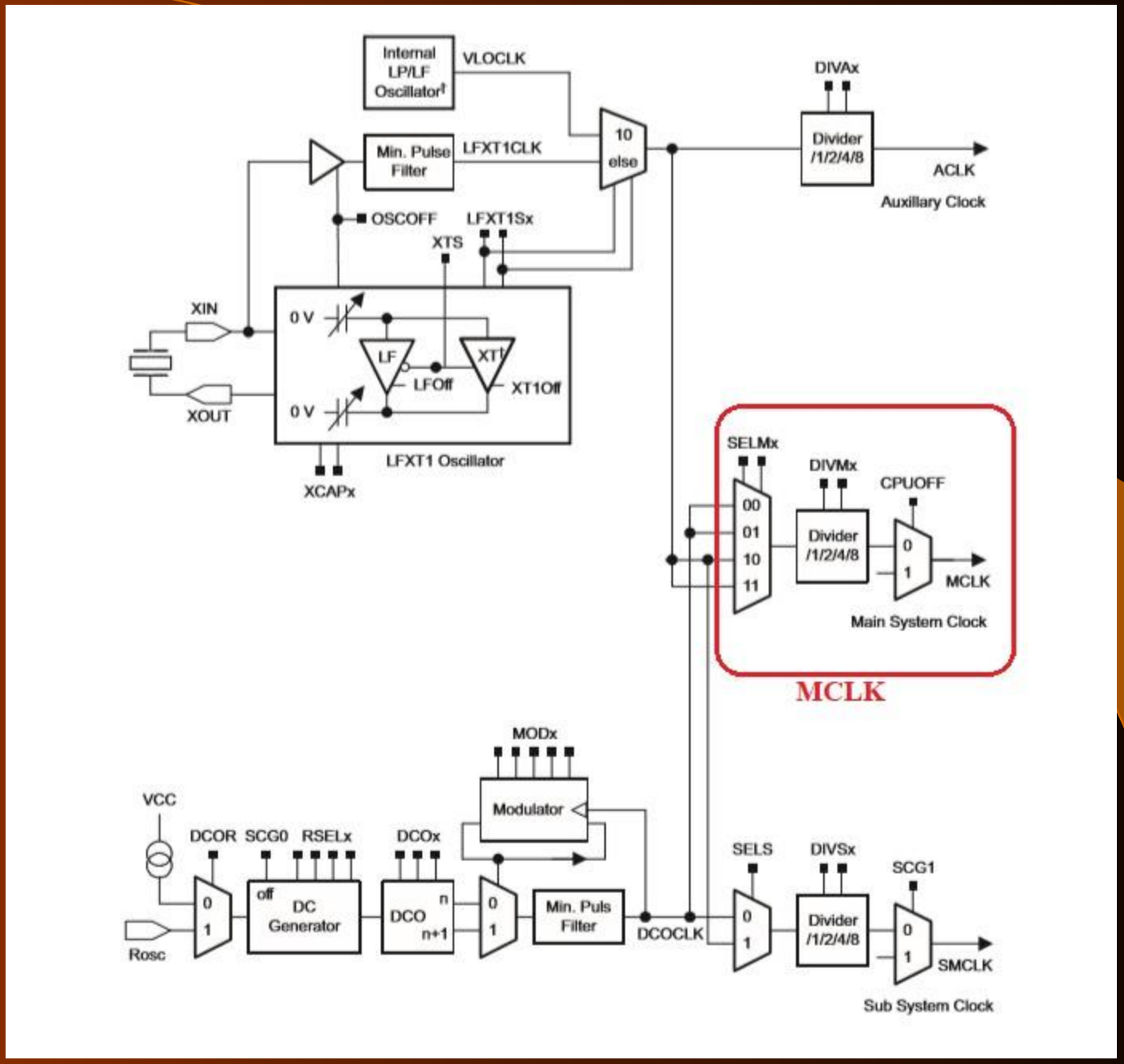
- Why do we need so many different sources of clock?
  Because of conflicting demands of high performance and low power.
  We need different kinds of clock for different purposes:
  • Low clock frequency for energy conservation and time keeping
  • High clock frequency for fast reaction to events and fast burst processing capability
  • Clock stability over operating temperature and supply voltage

- Using three internal clock signals, the user can select the best balance of performance and low power consumption.

Clock Signals

- **1. MCLK:** Master clock

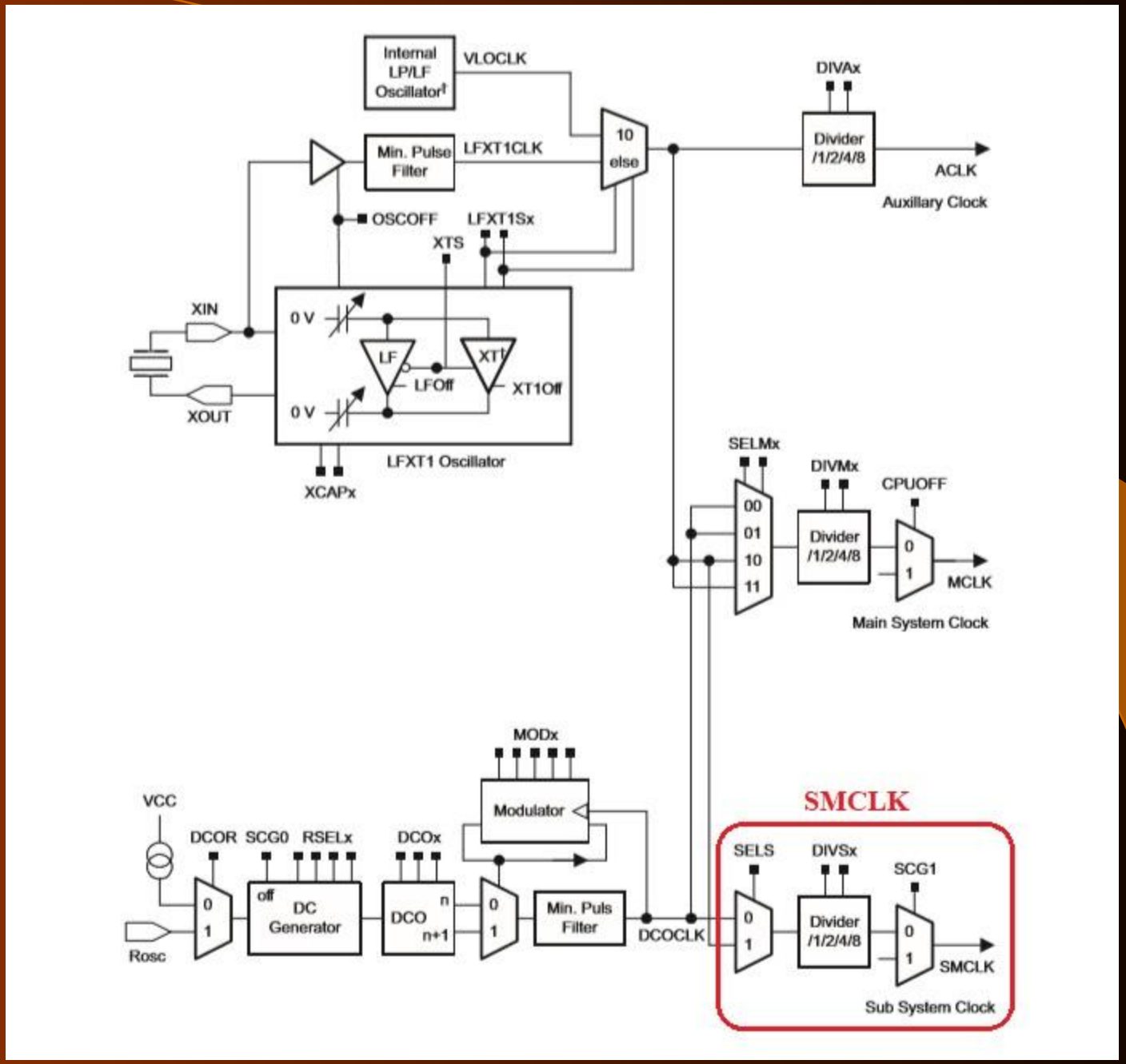
  - Used by the CPU and a few peripherals.
  - Supplied by the DCO with a frequency of around 1.1MHz. Stabilized by PLL.
  - MCLK is software selectable as LFXT1CLK, VLOCLK, DCOCLK (or XT2CLK, but not present on MSP430G2553).


Internal LP/LF Oscillator†
VLOCLK
DIVAx
Min. Pulse Filter
LFXT1CLK
10
else
Divider /1/2/4/8
ACLK
Auxillary Clock
OSCOFF
LFXT1Sx
XTS
XIN
0 V
LF
XT1
LFOff
XT1Off
XOUT
0 V
LFXT1 Oscillator
XCAPx
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
MCLK
MODx
Modulator
VCC
DCOR
SCG0
RSELx
DCOx
SELS
DIVSx
SCG1
0
1
off
DC Generator
DCO
n
n+1
Min. Puls Filter
DCOCLK
Divider /1/2/4/8
SMCLK
Rosc
Sub System Clock

Clock Signals

- **2. SMCLK:** Sub-system Master Clock

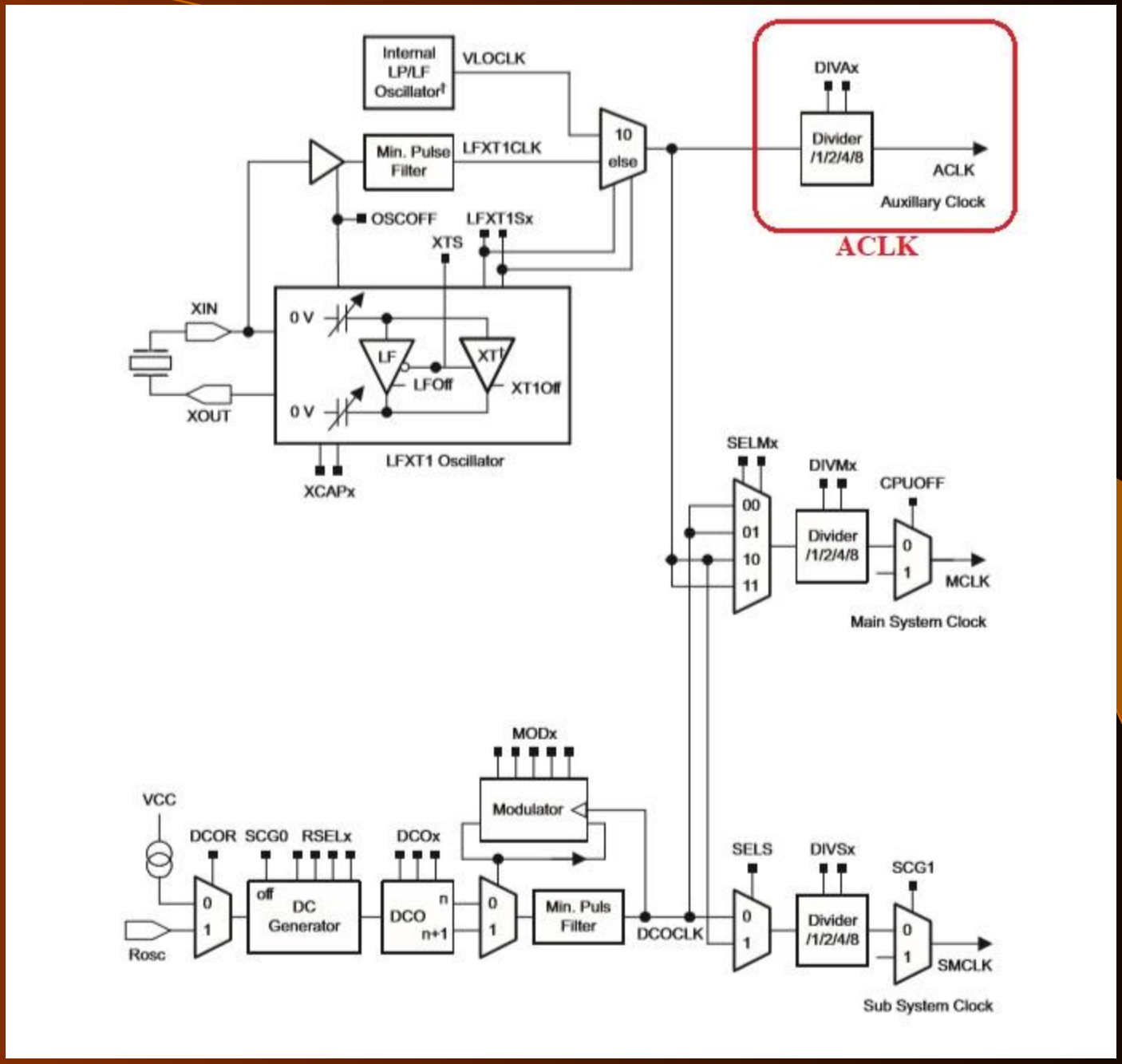
  - Distributed to peripherals.
  - Often same as MCLK.
  - Supplied by the DCO with a frequency of around 1.1MHz. Stabilized by PLL.
  - SMCLK is software selectable as LFXT1CLK, VLOCLK, DCOCLK (or XT2CLK, but not present on MSP430G2553).


Internal LP/LF Oscillator†
VLOCLK
Min. Pulse Filter
LFXT1CLK
10
else
DIVAx
Divider /1/2/4/8
ACLK
Auxiliary Clock
OSCOFF
LFXT1Sx
XTS
XIN
XOUT
0 V
0 V
LF
XT1
LFOff
XT1Off
LFXT1 Oscillator
XCAPx
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
MODx
Modulator
SMCLK
VCC
DCOR
SCG0
RSELx
DCOx
SELS
DIVSx
SCG1
Rosc
off
DC Generator
DCO
n
n+1
Min. Puls Filter
DCOCLK
Divider /1/2/4/8
SMCLK
Sub System Clock

Clock Signals

- **3. ACLK**: Auxiliary Clock

  - Distributed among peripherals.
  - Sourced from LFXT1CLK or VLOCLK.
  - Typically much slower and usually <=32kHz.
  - ACLK is software selectable as LFXT1CLK or VLOCLK.


Internal LP/LF Oscillator†
VLOCLK
Min. Pulse Filter
LFXT1CLK
10
else
DIVAx
Divider /1/2/4/8
ACLK
Auxillary Clock
ACLK
OSCOFF
LFXT1Sx
XTS
XIN
XOUT
0 V
0 V
LF
XT1
LFOff
XT1Off
LFXT1 Oscillator
XCAPx
SELMx
DIVMx
CPUOFF
00
01
10
11
Divider /1/2/4/8
0
1
MCLK
Main System Clock
MODx
Modulator
VCC
DCOR
SCG0
RSELx
DCOx
off
DC Generator
DCO
n
n+1
Min. Puls Filter
DCOCLK
SELS
DIVSx
SCG1
Divider /1/2/4/8
SMCLK
Sub System Clock
Rosc

Calibrated Frequencies of DCO

- 4 Calibrated frequencies:
  - 1 MHz
  - 8 MHz
  - 12 MHz
  - 16 MHz

- **Sample CCS Statements:**

  BCSCTL1 = CALBC1_1MHZ; (For range selection)

  DCOCTL = CALDCO_1MHZ; (For Freq. selection)

Wish to add crystal?

- Crystals are used when an accurate, stable frequency is needed.
- Crystals are cut from carefully grown, high-quality quartz with specific orientations to give them high stability.
- Traditional crystals oscillate at frequencies of a few MHz.
- Machined into complicated tuning fork shapes to give the low frequency.
- Designed to be most stable at 25 C.
- Crystal connected between XIN & XOUT.

Wish to add crystal?

Disadvantage:

- Frequency is more sensitive to temperature.
  SOLUTION => EXTERNAL OSCILLATOR
- Requires a **capacitance** to GND from each pin. Value depends on the crystal and is typically 10pF or 22pF.
- Part of it contributed by stray capacitance from PCB track, therefore kept short.
- External capacitance needed for many controllers/RTC ICs but integrated in MSP430.

Clock Registers

- **DCOCTL (DCO Control Register)**
- **BCSCTL1 (Basic Clock System Control Register 1)**
- **BCSCTL2 (Basic Clock System Control Register 2):** For MCLK & SMCLK manipulation
- **BCSCTL3 (Basic Clock System Control Register 3)**: For external crystal and capacitor selections

DCOCTL: DCO Control Register

7	6	5	4	3	2	1	0
DCOx			MODx				
rw-0	rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0

DCOx	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.
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-MOD) the f_{DCO} frequency is used. Not useable when DCOx = 7.

BCSCTL1: Basic Clock 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.

(1) XTS = 1 is not supported in MSP430x20xx and MSP430G2xx devices (see Figure 5-1 and Figure 5-2 for details on supported settings for all devices).

(2) This bit is reserved in the MSP430AFE2xx devices.

BCSCTL2: Basic Clock Control Register 2

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
		01	/2
		10	/4
		11	/8
DCOR	Bit 0	DCO resistor select. Not available in all devices. See the device-specific data sheet.	
		0	Internal resistor
		1	External resistor

(1) Does not apply to MSP430x20xx or MSP430x21xx devices.
(2) This bit is reserved in the MSP430AFE2xx devices.

BCSCTL3: Basic Clock Control Register 3

7	6	5	4	3	2	1	0
XT2Sx		LFXT1Sx[1]		XCAPx[2]		XT2OF[3]	LFXT1OF[2]
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 MSP430F21x1 devices)
- 11 Digital external clock source

When XTS = 1 (Not applicable for MSP430x20xx devices, MSP430G2xx1/2/3)

- 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 definition for MSP430AFE2xx devices:

- 00 Reserved
- 01 Reserved
- 10 VLOCLK
- 11 Reserved

XCAPx Bits 3-2 Oscillator capacitor selection. These bits select the effective capacitance seen by the LFXT1 crystal when XTS = 0. If XTS = 1 or if LFXT1Sx = 11 XCAPx should be 00.

- 00 ~1 pF
- 01 ~6 pF
- 10 ~10 pF
- 11 ~12.5 pF

XT2OF Bit 1 XT2 oscillator fault

- 0 No fault condition present
- 1 Fault condition present

LFXT1OF Bit 0 LFXT1 oscillator fault

- 0 No fault condition present
- 1 Fault condition present

(1) MSP430G22x0: The LFXT1Sx bits should be programmed to 10b during the initialization and start-up code to select VLOCLK (for more details refer to Digital I/O chapter). The other bits are reserved and should not be altered.

(2) This bit is reserved in the MSP430AFE2xx devices.

(3) Does not apply to MSP430x2xx, MSP430x21xx, or MSP430x22xx devices.

Setting the frequency of the DCO

The frequency of DCOCLK is set by the following functions:

- The four RSELx bits select one of sixteen nominal frequency ranges for the DCO. These ranges are defined in the datasheet.
- The three DCOx bits divide the DCO range selected by the RSELx bits into 8 frequency steps, separated by approximately 10%.
- The five MODx bits, switch between the frequency selected by the DCOx bits and the next higher frequency set by DCOx+1. When DCOx = 07h, the MODx bits have no effect because the DCO is already at the highest setting for the selected RSELx range.

Setting the frequency of the DCO

DCO Frequency

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	V_{CC}	MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage	RSELx < 14		1.8		3.6	V
		RSELx = 14		2.2		3.6	
		RSELx = 15		3		3.6	
$f_{DCO(0,0)}$	DCO frequency (0, 0)	RSELx = 0, DCOx = 0, MODx = 0	3 V	0.06		0.14	MHz
$f_{DCO(0,3)}$	DCO frequency (0, 3)	RSELx = 0, DCOx = 3, MODx = 0	3 V	0.07		0.17	MHz
$f_{DCO(1,3)}$	DCO frequency (1, 3)	RSELx = 1, DCOx = 3, MODx = 0	3 V		0.15		MHz
$f_{DCO(2,3)}$	DCO frequency (2, 3)	RSELx = 2, DCOx = 3, MODx = 0	3 V		0.21		MHz
$f_{DCO(3,3)}$	DCO frequency (3, 3)	RSELx = 3, DCOx = 3, MODx = 0	3 V		0.30		MHz
$f_{DCO(4,3)}$	DCO frequency (4, 3)	RSELx = 4, DCOx = 3, MODx = 0	3 V		0.41		MHz
$f_{DCO(5,3)}$	DCO frequency (5, 3)	RSELx = 5, DCOx = 3, MODx = 0	3 V		0.58		MHz
$f_{DCO(6,3)}$	DCO frequency (6, 3)	RSELx = 6, DCOx = 3, MODx = 0	3 V	0.54		1.06	MHz
$f_{DCO(7,3)}$	DCO frequency (7, 3)	RSELx = 7, DCOx = 3, MODx = 0	3 V	0.80		1.50	MHz
$f_{DCO(8,3)}$	DCO frequency (8, 3)	RSELx = 8, DCOx = 3, MODx = 0	3 V		1.6		MHz
$f_{DCO(9,3)}$	DCO frequency (9, 3)	RSELx = 9, DCOx = 3, MODx = 0	3 V		2.3		MHz
$f_{DCO(10,3)}$	DCO frequency (10, 3)	RSELx = 10, DCOx = 3, MODx = 0	3 V		3.4		MHz
$f_{DCO(11,3)}$	DCO frequency (11, 3)	RSELx = 11, DCOx = 3, MODx = 0	3 V		4.25		MHz
$f_{DCO(12,3)}$	DCO frequency (12, 3)	RSELx = 12, DCOx = 3, MODx = 0	3 V	4.30		7.30	MHz
$f_{DCO(13,3)}$	DCO frequency (13, 3)	RSELx = 13, DCOx = 3, MODx = 0	3 V	6.00	7.8	9.60	MHz
$f_{DCO(14,3)}$	DCO frequency (14, 3)	RSELx = 14, DCOx = 3, MODx = 0	3 V	8.60		13.9	MHz
$f_{DCO(15,3)}$	DCO frequency (15, 3)	RSELx = 15, DCOx = 3, MODx = 0	3 V	12.0		18.5	MHz
$f_{DCO(15,7)}$	DCO frequency (15, 7)	RSELx = 15, DCOx = 7, MODx = 0	3 V	16.0		26.0	MHz
S_{RSEL}	Frequency step between range RSEL and RSEL+1	$S_{RSEL} = f_{DCO(RSEL+1,DCO)}/f_{DCO(RSEL,DCO)}$	3 V		1.35		ratio
S_{DCO}	Frequency step between tap DCO and DCO+1	$S_{DCO} = f_{DCO(RSEL,DCO+1)}/f_{DCO(RSEL,DCO)}$	3 V		1.08		ratio
	Duty cycle	Measured at SMCLK output	3 V		50		%

Clock Code Example

```
#include <msp430.h>

#define LED BIT7

#define SW1 BIT3                        // 1.5 kHz
#define SW2 BIT4                        //   3 kHz
#define SW3 BIT5                        //  12 kHz

volatile unsigned int i;                // Volatile to prevent removal

/**
 * @brief
 * Function to take input from 3 switches and change CPU clock accordingly
 **/
void switch_input()
{

    if(!(P1IN & SW1))           // If SW1 is Pressed
    {
        __delay_cycles(2000);   // Wait 20ms to debounce
        while(!(P1IN & SW1));   // Wait till SW Released
        __delay_cycles(2000);   // Wait 20ms to debounce

        BCSCTL2 &=~ (BIT5 + BIT4);      //Reset VLO divider
        BCSCTL2 |= (BIT5 + BIT4);       //VLO = 12kHz/8 = 1.5kHz
    }


    if(!(P1IN & SW2))           // If SW is Pressed
    {
        __delay_cycles(2000);   // Wait 20ms to debounce
        while(!(P1IN & SW2));   // Wait till SW Released
        __delay_cycles(2000);   // Wait 20ms to debounce

        BCSCTL2 &=~ (BIT5 + BIT4);      //Reset VLO divider
        BCSCTL2 |= (BIT4);              //VLO = 12kHz/4 = 3kHz
    }

```

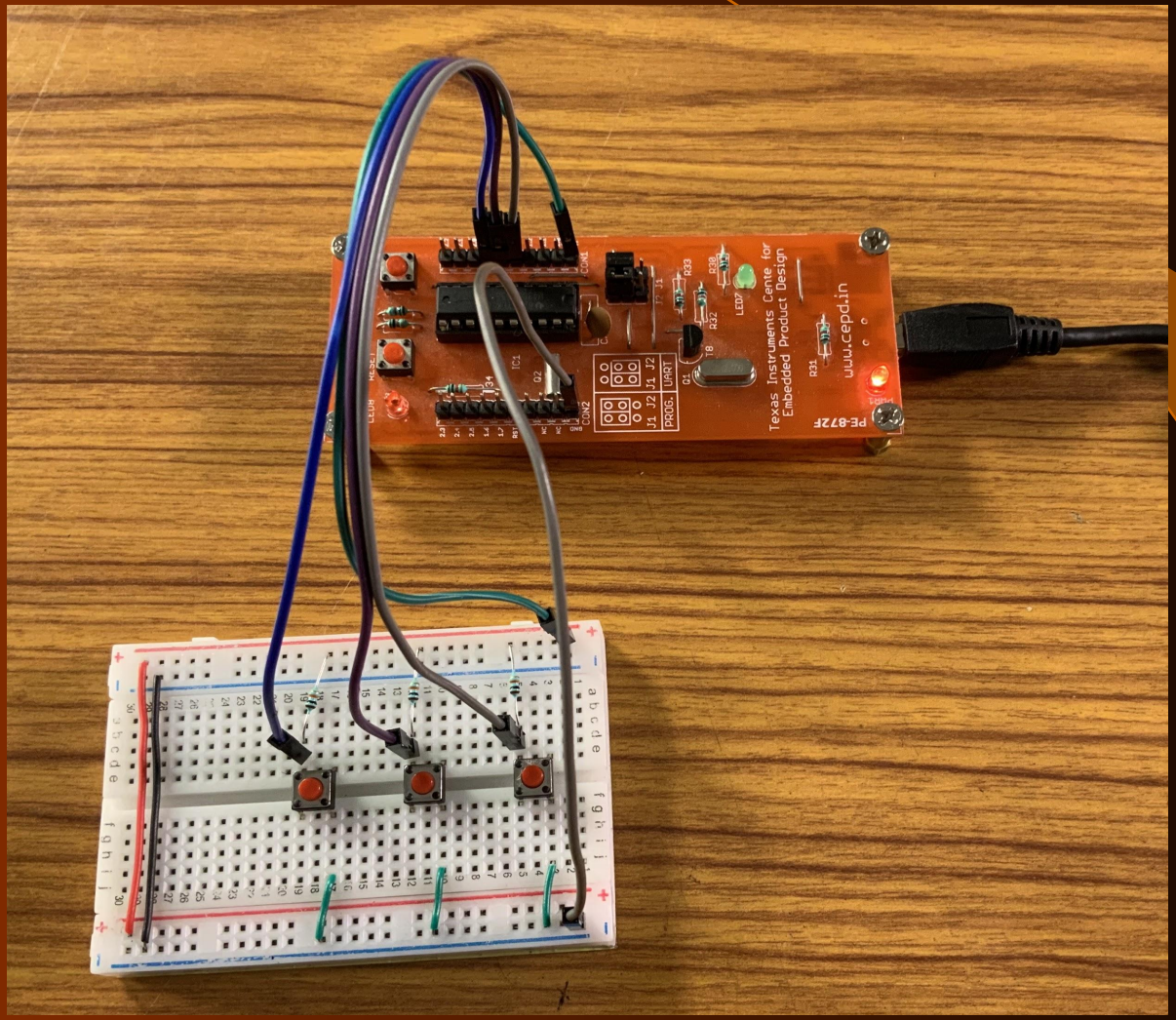
Clock Code Example

```
39      if(!(P1IN & SW3))            // If SW is Pressed
40      {
41
42          __delay_cycles(2000);    // Wait 20ms to debounce
43          while(!(P1IN & SW3));    // Wait till SW Released
44          __delay_cycles(2000);    // Wait 20ms to debounce
45
46          BCSCTL2 &=~ (BIT5 + BIT4);      //VLO = 12kHz/1 = 12kHz
47      }
48
49  }
50
51  /**
52   * @brief
53   * These settings are wrt enabling GPIO on Lunchbox
54   **/
55  void register_settings_for_GPIO()
56  {
57      P1DIR |= LED;                       //P1.7 is set as Output
58      P1DIR &=~ (SW1 + SW2 + SW3);        //P1.3, P1.4, P1.5 are set as Input
59  }
60
61
62  /**
63   * @brief
64   * These settings are w.r.t enabling VLO  on Lunch Box
65   **/
66  void register_settings_for_VLO()
67  {
68      BCSCTL3 |= LFXT1S_2;                        // LFXT1 = VLO
69         do{
70             IFG1 &= ~OFIFG;                      // Clear oscillator fault flag
71             for (i = 50000; i; i--);             // Delay
72           } while (IFG1 & OFIFG);                // Test osc fault flag
73
74         BCSCTL2 |= SELM_3;                       // MCLK = VLO
75  }
76
```

Clock Code Example

```
76
77 /**
78  * main.c
79  */
80 int main(void)
81 {
82
83     WDTCTL = WDTPW | WDTHOLD;                    //Stop watchdog timer
84
85     register_settings_for_VLO();                 //Register settings for VLO
86     register_settings_for_GPIO();                //Register settings for GPIO
87
88     while(1)
89     {
90         switch_input();                          //Switch Input
91
92         P1OUT ^= LED;                            //LED Toggle
93         for (i = 100; i > 0; i--);
94
95     }
96 }
```

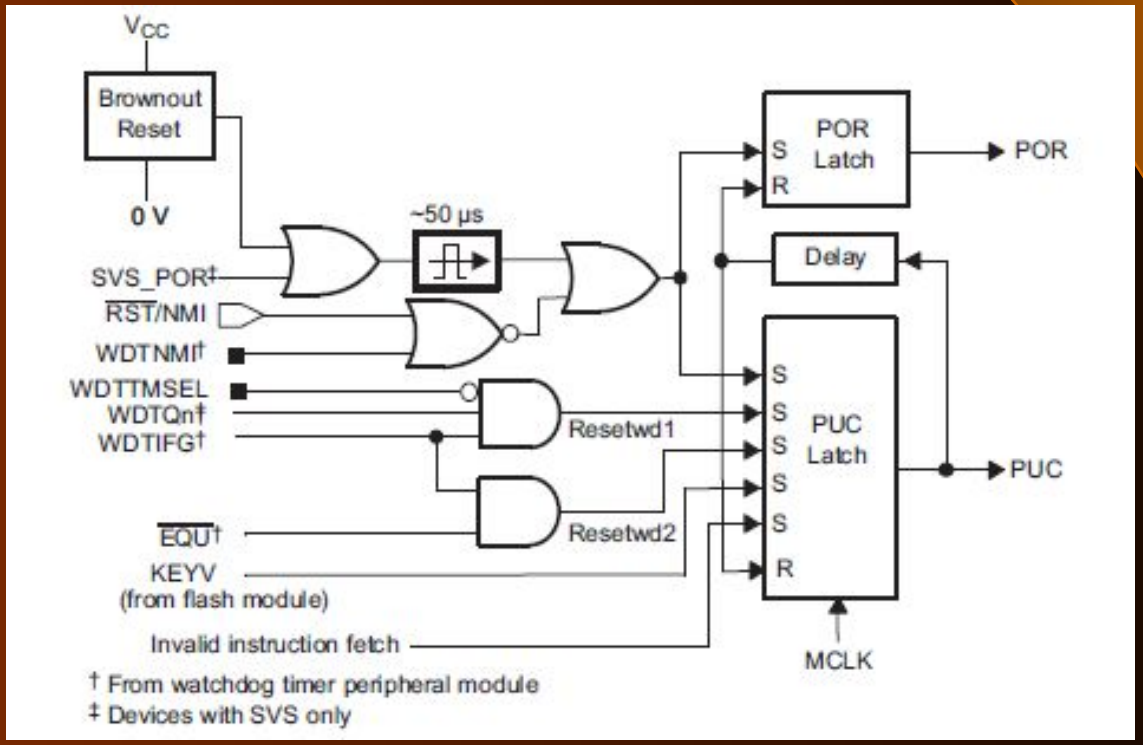
Circuit For The Clock Source Example Code



MSP430 Resets

There are two types of resets:

1. Power on reset (POR)
2. Power up clear (PUC)



Power On Reset

A POR is a device reset and is generated by conditions related to hardware. It occurs in any of the following conditions:

1. Powering up the device
2. Brownout Reset: A POR is raised even when the supply voltage drops to so low a value that the device may not work correctly.
3. A high to low signal on the RST/NMI pin when configured in the reset mode. (User Reset)

Power Up Clear

A PUC is generated by software conditions. It is also always generated when a POR is generated, but a POR is not generated by a PUC. The following events trigger a PUC:

1. A POR signal
2. Watchdog timer expiration when in watchdog mode only
3. Watchdog timer security key violation (An attempt is made to write to the watchdog control register WDTCTL without the correct password 0x05A)
4. A Flash memory security key violation
5. A CPU instruction fetch from the peripheral address range of 0000h to 01FFh

Device Initial Conditions After System Reset

After a POR, the initial MSP430 conditions are:

- The RST/NMI pin is configured in the reset mode.
- I/O pins are switched to input mode.
- Other peripheral modules and registers are initialized as described in the user guide.

  The value is shown under each bit in the description of the registers. For example, rw–0 means that a bit can be both read and written and is initialized to 0 after a PUC. Whereas, rw–(0) shows that a bit is initialized to 0 only after a POR; it retains its value through a PUC.

Device Initial Conditions After System Reset

- Status register (SR) is reset. This means that the device operates at full power, even though it might have been in a low-power mode before the reset occurred.
- The watchdog timer powers up active in watchdog mode.
- Program counter (PC) is loaded with address contained at reset vector location (FFFEh).

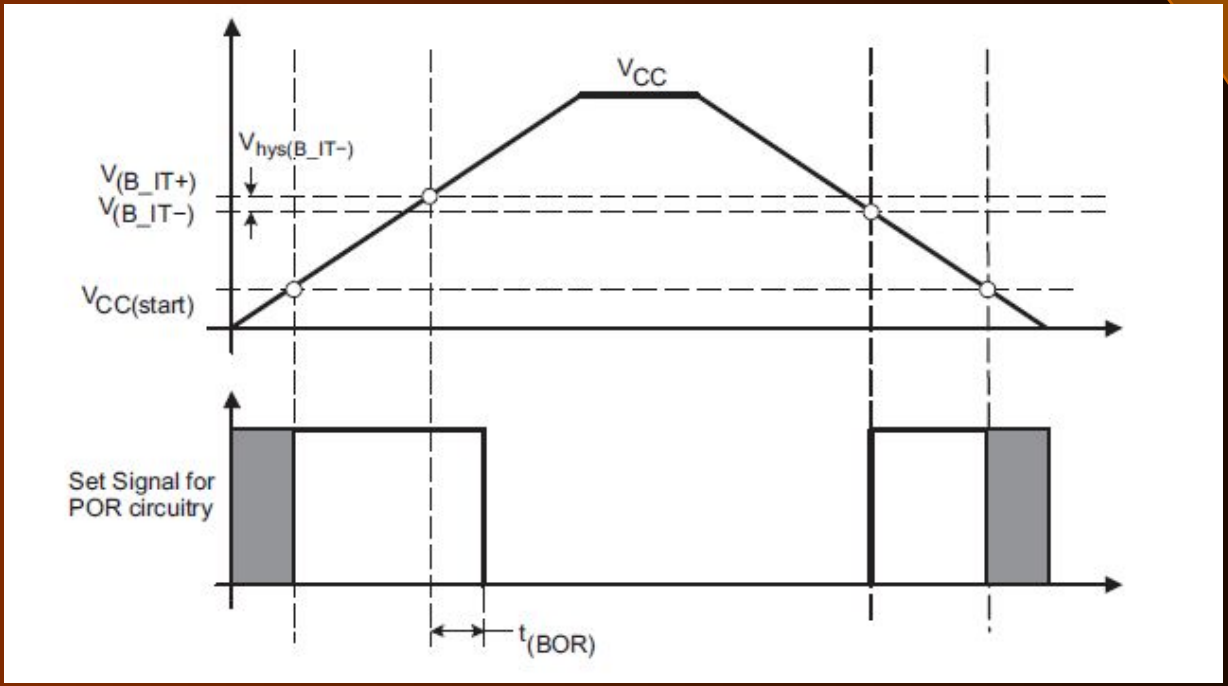
Software Initialisations after System Reset

After a system reset, the software must:

- Initialize the watchdog timer, usually to turn it off
- Configure peripheral modules

Brownout Reset

- The brownout reset circuit detects low supply voltages such as when a supply voltage is applied to or removed from the VCC terminal.
- The brownout reset circuit resets the device by triggering a POR signal when power is applied or removed.



Brownout Reset

- The POR signal becomes active when VCC crosses the VCC(start) level.
- It remains active until VCC crosses the V(B_IT+) threshold and the delay t(BOR) elapses.
- The delay t(BOR) is adaptive being longer for a slow ramping VCC. The hysteresis Vhys(B_ IT-) ensures that the supply voltage must drop below V(B_IT-) to generate another POR signal from the brownout reset circuitry.

Watchdog Timer

- The primary function of the watchdog timer (WDT+) module is to perform a controlled system restart after a software problem occurs.
- If the selected time interval expires, a system reset is generated.
- If the watchdog function is not needed in an application, the module can be disabled or configured as an interval timer and can generate interrupts at selected time intervals.

Watchdog Timer Operating Modes

The watchdog timer+ module has two operating modes:

1. Watchdog mode:

- After a PUC, the WDT+ module is configured in the watchdog mode with an initial 32768 cycle reset interval using the DCOCLK.
- The user must setup, halt, or clear the WDT+ prior to the expiration of the initial reset interval or another PUC will be generated.
- When the WDT+ is configured to operate in watchdog mode, either writing to WDTCTL with an incorrect password, or expiration of the selected time interval triggers a PUC.

Watchdog Timer Operating Modes

2. Interval mode

- This mode can be used to provide periodic interrupts.
- In interval timer mode, the WDTIFG flag is set at the expiration of the selected time interval.
- A PUC is not generated in interval timer mode at expiration of the selected timer interval and the WDTIFG enable bit WDTIE remains unchanged.
- When the WDTIE bit and the GIE bit are set, the WDTIFG flag requests an interrupt. The WDTIFG interrupt flag is automatically reset when its interrupt request is serviced, or may be reset by software. The interrupt vector address in interval timer mode is different from that in watchdog mode.

Watchdog Timer Registers

- WDTCTL: Watchdog Timer+ Control Register:
- IE1: SFR Interrupt Enable Register 1
- IFG1: SFR Interrupt Flag Register 1

WDTCTL Register

- WDTCTL is a 16-bit, password-protected, read/write register.
- Any read or write access must use word instructions and write accesses must include the write password 05Ah in the upper byte.
- Any write to WDTCTL with any value other than 05Ah in the upper byte is a security key violation and triggers a PUC system reset regardless of timer mode.
- Any read of WDTCTL reads 069h in the upper byte.

WDTCTL Register

15	14	13	12	11	10	9	8
WDTPW, Read as 069h Must be written as 05Ah							

7	6	5	4	3	2	1	0
WDTHOLD	**WDTNMIES**	**WDTNMI**	**WDTTMSEL**	**WDTCNTCL**	**WDTSSEL**	**WDTISx**	
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-0	rw-0	rw-0

WDTPW Bits 15-8 Watchdog timer+ password. Always read as 069h. Must be written as 05Ah, or a PUC is generated.

WDTHOLD Bit 7 Watchdog timer+ hold. This bit stops the watchdog timer+. Setting WDTHOLD = 1 when the WDT+ is not in use conserves power.

- 0 Watchdog timer+ is not stopped
- 1 Watchdog timer+ is stopped

WDTNMIES Bit 6 Watchdog timer+ NMI edge select. This bit selects the interrupt edge for the NMI interrupt when WDTNMI = 1. Modifying this bit can trigger an NMI. Modify this bit when WDTIE = 0 to avoid triggering an accidental NMI.

- 0 NMI on rising edge
- 1 NMI on falling edge

WDTNMI Bit 5 Watchdog timer+ NMI select. This bit selects the function for the $\overline{\text{RST}}$/NMI pin.

- 0 Reset function
- 1 NMI function

WDTTMSEL Bit 4 Watchdog timer+ mode select

- 0 Watchdog mode
- 1 Interval timer mode

WDTCNTCL Bit 3 Watchdog timer+ counter clear. Setting WDTCNTCL = 1 clears the count value to 0000h. WDTCNTCL is automatically reset.

- 0 No action
- 1 WDTCNT = 0000h

WDTSSEL Bit 2 Watchdog timer+ clock source select

- 0 SMCLK
- 1 ACLK

WDTISx Bits 1-0 Watchdog timer+ interval select. These bits select the watchdog timer+ interval to set the WDTIFG flag and/or generate a PUC.

- 00 Watchdog clock source /32768
- 01 Watchdog clock source /8192
- 10 Watchdog clock source /512
- 11 Watchdog clock source /64

IE1: SFR Interrupt Enable Register 1

Address	7	6	5	4	3	2	1	0
00h			ACCVIE	NMIIE			OFIE	WDTIE
			rw-0	rw-0			rw-0	rw-0

WDTIE	Watchdog Timer interrupt enable. Inactive if watchdog mode is selected. Active if Watchdog Timer is configured in interval timer mode.
OFIE	Oscillator fault interrupt enable
NMIIE	(Non)maskable interrupt enable
ACCVIE	Flash access violation interrupt enable

- WDTIE bit enables the WDTIFG interrupt for interval timer mode. It is not necessary to set this bit for watchdog mode.

IFG1: SFR Interrupt Flag Register 1

Address	7	6	5	4	3	2	1	0
02h				NMIIFG	RSTIFG	PORIFG	OFIFG	WDTIFG
				rw-0	rw-(0)	rw-(1)	rw-1	rw-(0)

WDTIFG	Set on watchdog timer overflow (in watchdog mode) or security key violation. Reset on V_{CC} power-on or a reset condition at the $\overline{RST}$/NMI pin in reset mode.
OFIFG	Flag set on oscillator fault.
PORIFG	Power-On Reset interrupt flag. Set on V_{CC} power-up.
RSTIFG	External reset interrupt flag. Set on a reset condition at $\overline{RST}$/NMI pin in reset mode. Reset on V_{CC} power-up.
NMIIFG	Set via $\overline{RST}$/NMI pin

How to stop the watchdog timer?

The watchdog timer is stopped by the following statement:
WDTCTL = WDTPW + WDTHOLD;

This instruction sets the password (WDTPW) as 5Ah and the bit to stop the timer (WDTHOLD).

Determining the Reset Source: Code Example

```c
#include <msp430.h>

#define  LED1 BIT0                                  // main() has started
#define  LED2 BIT1                                  // program is executing
#define  LEDA BIT2                                  // POR
#define  LEDB BIT3                                  // WDT
#define  LEDC BIT4                                  // RST

/**
 * @brief
 * These settings are wrt enabling GPIO on Lunchbox
 **/
void register_settings_for_GPIO()
{
    P1DIR |= (LED1 + LED2 + LEDA + LEDB + LEDC);        // P1.0 to P1.4 as output

    P1OUT |= LED1;                                  // LED1 representing main() is set

    P1OUT &=~ (LED2 + LEDA + LEDB + LEDC);      // Turning all other LEDs OFF
}

/**
 * @brief
 * These settings are w.r.t check source of reset on Lunch Box
 **/
void checking_reset_source()
{
    if (IFG1 & PORIFG)                          // Check for Power on Reset flag (POR)
    {
        P1OUT|= LEDA;                           // Turning LEDA On
        P1OUT &= ~LEDB;                         // Turning LEDB Off
        P1OUT &= ~LEDC;                         // Turning LEDC Off

        /* Settings for Watchdog Timer register
                Giving Watchdog Timer Password
                Clearing Watchdog counter to 0000h
                Watchdog Source select --> ACLK
                Watchdog Timer interval set to generate watchdog reset at 2 secs
        */
        WDTCTL = (WDTPW + WDTCNTCL + WDTSSEL + WDTIS0);

        IFG1 &=~ PORIFG;                        // Clearing Power on Reset flag
    }
```

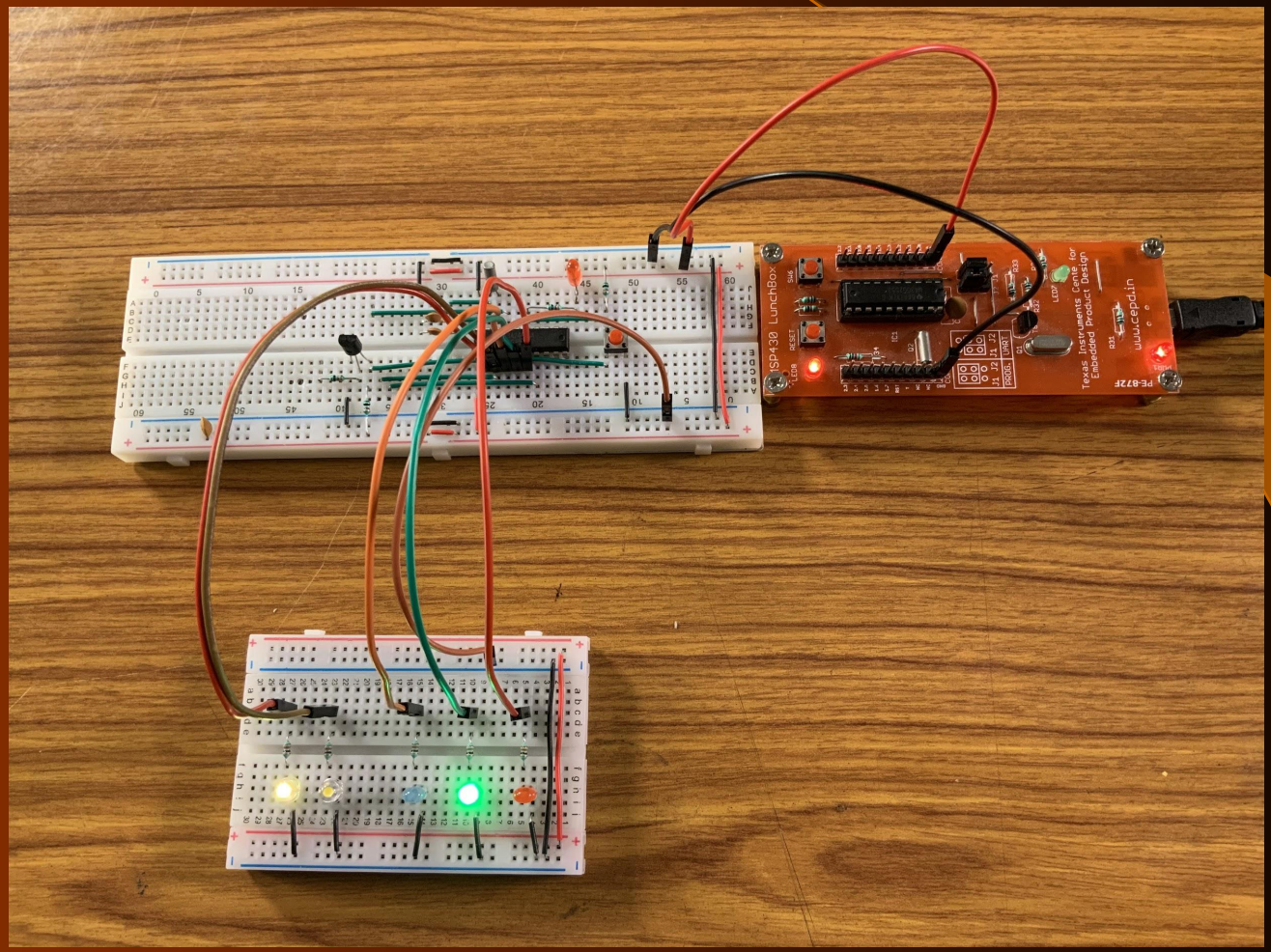
Determining the Reset Source: Code Example

```
43      }
44
45      else if (IFG1 & RSTIFG)                 // Check for External Reset flag (RST)
46      {
47          P1OUT|= LEDC;                       // Turning LEDC On
48          P1OUT &= ~LEDA;                     // Turning LEDA Off
49          P1OUT &= ~LEDB;                     // Turning LEDB Off
50
51          /* Settings Watchdog Timer register
52                  Giving Watchdog Timer Password
53                  Clearing Watchdog counter to 0000h
54                  Watchdog Source select --> ACLK
55                  Watchdog Timer interval set to generate watchdog reset at 2 secs
56          */
57          WDTCTL = (WDTPW + WDTCNTCL + WDTSSEL + WDTIS0);
58
59          IFG1 &=~ RSTIFG;                    // Clearing External Reset flag
60      }
61
62      else if (IFG1 & WDTIFG)                 // Check for Watch Dog Reset flag (WDT)
63      {
64          P1OUT|= LEDB;                       // Turning LEDB On
65          P1OUT &= ~LEDA;                     // Turning LEDA Off
66          P1OUT &= ~LEDC;                     // Turning LEDC Off
67
68          IFG1 &=~ PORIFG;                    // Clearing Power on Reset flag
69          IFG1 &=~ RSTIFG;                    // Clearing External Reset flag
70      }
71
72  }
```

Determining the Reset Source: Code Example

```
73
74 /*@brief entry point for the code*/
75 void main(void)
76 {
77       WDTCTL = WDTPW | WDTHOLD;    // stop watch dog timer
78       volatile unsigned int i;
79
80       BCSCTL1 |= DIVA_3;             // Dividing ACLK by 8, 32768Hz/8 = 4096Hz
81
82       register_settings_for_GPIO();
83
84       /* This loop checks for Oscillator fault flag to reset means
85          it delays execution until external crystal is Power On  */
86
87       do
88       {
89          IFG1 &= ~OFIFG;                          // Clear oscillator fault flag
90          for (i = 10000; i ; i--);                // Delay, minimum value of i = 5000
91       }  while (IFG1 & OFIFG);                    // Test osc fault flag
92
93       checking_reset_source();
94
95       while(1)
96       {
97           P1OUT ^= LED2;                         // Toggle LED
98           __delay_cycles(1000000);
99       }
100 }
101
```

Circuit For The Reset Source Example Code



Thank you!