

CPE/EE 421 Microcomputers

Instructor: Dr Aleksandar Milenkovic
Lecture Note
S13

*Material used is in part developed by
Dr. D. Raskovic and Dr. E. Jovanov

MSP430 Documentation

- MSP430 home page (TI)
 - ❖ www.ti.com/msp430
- User's manual
 - ❖ <http://www.ece.uah.edu/~milenka/cpe421-04S/manuals/slau049c.pdf>
- Datasheet
 - ❖ <http://www.ece.uah.edu/~milenka/cpe421-04S/manuals/slas272c.pdf>
- TI Workshop document
 - ❖ http://www.ece.uah.edu/~milenka/cpe421-04S/manuals/430_2002_atc_workshop.pdf
- IAR Workbench Tutorial
 - ❖ <http://www.ece.uah.edu/~milenka/cpe421-04S/manuals/TUTOR.pdf>

The MSP430 Clock Module



Basic Clock Systems

MSP430 Clock System

- ❖ **Low System Cost**
- ❖ **Low Power**

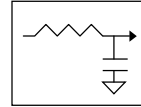
- Variety of operating modes driven by application, software selectable
- Support for the *Burst Mode* - when activated system starts and reacts rapidly
- Stability over voltage and temperature

Basic Clock Systems

➤ Basic Clock System- MSP430x1xx

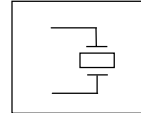
➤ One DCO, digital controlled oscillator

Generated on-chip
RC-type
frequency controlled by SW + HW



➤ One LF/XT oscillator

LF: 32768Hz
XT: 450kHz 8MHz



➤ Second LF/XT2 oscillator

➤ Clocks:

ACLK auxiliary clock ACLK
MCLK main system clock MCLK
SMCLK sub main system clock

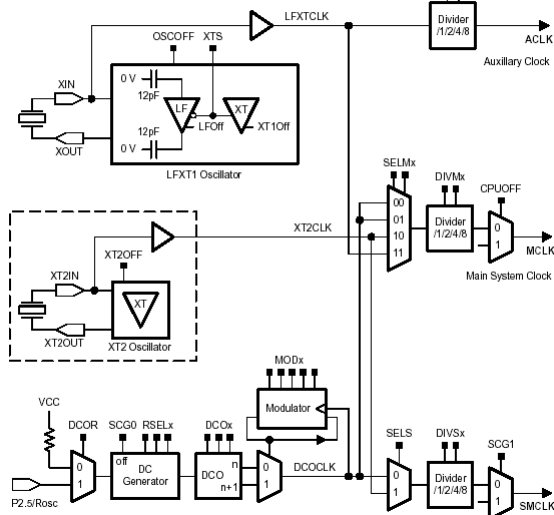
Clock sources

- **LFXT1CLK**: Low-frequency/high-frequency oscillator that can be used either with low-frequency 32,768-Hz watch crystals, or standard crystals, resonators, or external clock sources in the 450-kHz to 8-MHz range.
- **XT2CLK**: Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 450-kHz to 8-MHz range.
- **DCOCLK**: Internal digitally controlled oscillator (DCO) with RC-type characteristics.

Available clocks

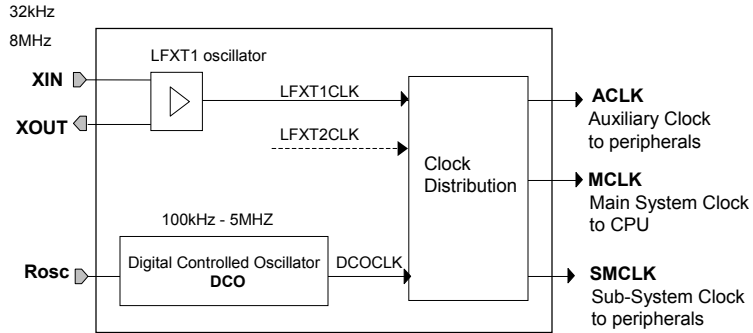
- **ACLK**: Auxiliary clock. The ACLK is the buffered LFXT1CLK clock source divided by 1, 2, 4, or 8. ACLK is software selectable for individual peripheral modules.
- **MCLK**: Master clock. MCLK is software selectable as LFXT1CLK, XT2CLK (if available), or DCOCLK. MCLK is divided by 1, 2, 4, or 8. MCLK is used by the CPU and system.
- **SMCLK**: Sub-main clock. SMCLK is software selectable as LFXT1CLK, XT2CLK (if available), or DCOCLK. SMCLK is divided by 1, 2, 4, or 8. SMCLK is software selectable for individual peripheral modules.

Basic clock block diagram

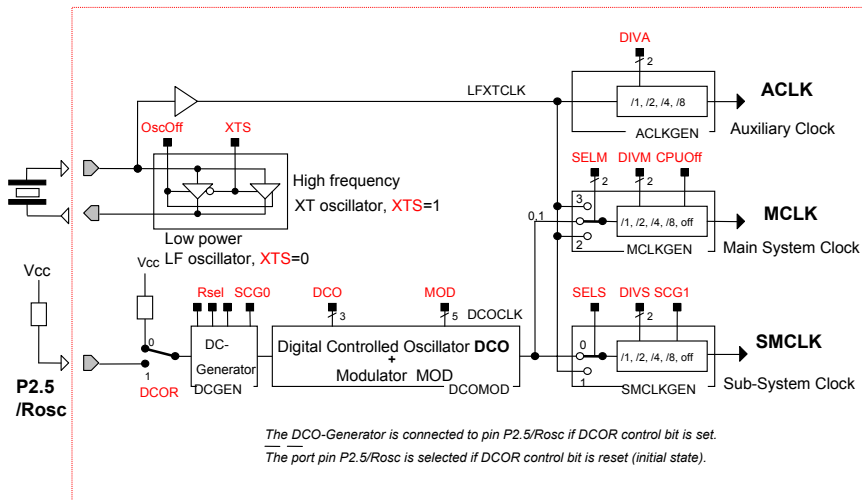


Basic Clock Systems

- DCOCLK Generated on-chip with 6 μ s start-up
- 32KHz Watch Crystal - or - High Speed Crystal / Resonator to 8MHz
 - ❖ (our system is 4MHz/8MHz high Speed Crystal)
- Flexible clock distribution tree for CPU and peripherals
- Programmable open-loop DCO Clock with internal and external current source



Basic Clock Systems-detail



Basic operation

- After POC (Power Up Clear) MCLK and SCLK are sourced by DCOCLK (approx. 800KHz) and ACLK is sourced by LFXT1 in LF mode
- Status register control bits **SCG0**, **SCG1**, **OSCOFF**, and **CPUOFF** configure the MSP430 operating modes and enable or disable portions of the basic clock module. The **DCOCTL**, **BCSCTL1**, and **BCSCTL2** registers configure the basic clock module
- The basic clock can be configured or reconfigured by software at any time during program execution

Low-power operation: An example

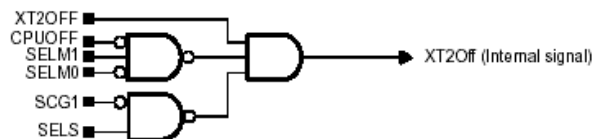
- ACLK can be configured to oscillate with a low-power 32,786-Hz watch crystal
- MCLK can be configured to operate from the on-chip DCO that can be only activated when requested by interrupt-driven events.
- SMCLK can be configured to operate from either the watch crystal or the DCO, depending on peripheral requirements.

LFXT1 Oscillator

- LF mode: XTS = 0
 - ❖ 32,768-Hz watch crystal in LF mode. A watch crystal connects to XIN and XOUT without any other external components.
- HF mode: XTS = 1
 - ❖ The high-speed crystal or resonator connects to XIN and XOUT and requires external capacitors on both terminals. These capacitors should be sized according to the crystal or resonator specifications.
- Software can disable LFXT1 by setting **OSCOFF**, if this signal does not source SMCLK or MCLK

XT2 Oscillator

- Similar to LFXT1 in HF mode
- **XT2OFF** bit disables the XT2 oscillator if XT2CLK is not used for MCLK or SMCLK

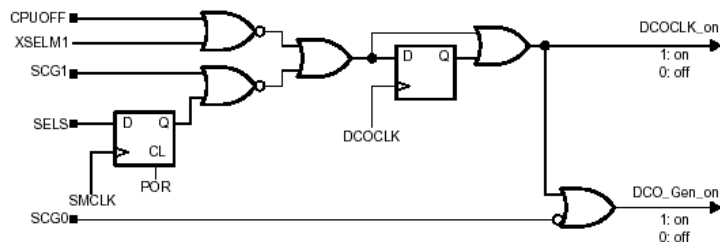


Digitally-Controlled Oscillator DCO

- Integrated ring oscillator with RC-type characteristics.
 - ❖ frequency varies with temperature, voltage, and from device to device.
- DCO frequency can be adjusted by software using the **DCOx**, **MODx**, and **RSELx** bits.
- The digital control of the oscillator allows frequency stabilization despite its RC-type characteristics.

Disabling DCO

- Software can disable DCO if not used for MCLK and SMCLK



Adjusting frequency

- After a PUC, an internal resistor is selected for the DC generator **RSELx=4**, and **DCOx=3**, allowing the DCO to start at a mid-range frequency.
- MCLK and SMCLK are sourced from DCOCLK. Because the CPU executes code from MCLK, which is sourced from the fast-starting DCO, code execution begins from PUC in less than 6 μ s.

Adjusting frequency

- DCO frequency is determined by:
 - ❖ The current injected into the DC generator by either the internal or external resistor defines the fundamental frequency.
 - **DCOR** bit selects the internal or external resistor.
 - ❖ The three **RSELx** bits select one of eight nominal frequency ranges for the DCO.
 - ❖ The three **DCOx** bits divide the DCO range into 8 frequency steps, separated by approx. 10%.
 - ❖ The five **MODx** bits, switch between the frequency selected by the **DCOx** bits and the next higher frequency set by **DCO+1**.

Basic Clock Module Registers

Table 4–1. Basic Clock Module Registers

Register	Short Form	Register Type	Address	Initial State
DCO control register	DCOCTL	Read/write	056h	056h with PUC
Basic clock system control 1	BCSCTL1	Read/write	057h	084h with PUC
Basic clock system control 2	BCSCTL2	Read/write	058h	Reset with POR
SFR interrupt enable register 1	IE1	Read/write	0000h	Reset with PUC
SFR interrupt flag register 1	IFG1	Read/write	0002h	Reset with PUC

Basic Clock Systems-control registers

- Direct SW Control
- DCOCLK can be Set - Stabilized
- Stable DCOCLK over Temp/Vcc.

BCSCTL2

058h

SELM.1	SELM.0	DIVM.1	DIVM.0	SELS	DIVS.1	DIVS.0	DCOR
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

BCSCTL1

057h

XT2Off	XTS	DIVA.1	DIVA.0	XT5V	Rsel.2	Rsel.1	Rsel.0
rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-1	rw-0	rw-0

Selection of DCO nominal frequency

DCOCTL

056h

DCO.2	DCO.1	DCO.0	MOD.4	MOD.3	MOD.2	MOD.1	MOD.0
rw-0	rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0

Which of eight discrete DCO frequencies is selected

Define how often frequency f_{DCO+1} within the period of 32 DCOCLK cycles is used. Remaining clock cycles (32-MOD) the frequency f_{DCO} is mixed

RSEL.x Select DCO nominal frequency
DCO.x and **MOD.x** set exact DCOCLK
 ... select other clock tree options

Basic Clock Systems-control registers(detail)

➤ Basic Clock Module Control Registers

The Basic Clock Module is configured using control registers DCOCTL, BCSCCTL1, and BCSCCTL2, and four bits from the CPU status register: SCG1, SCG0, OscOff, and CPUOFF.

User software can modify these control registers from their default condition at any time. The Basic Clock Module control registers are located in the byte-wide peripheral map and should be accessed with byte (.B) instructions.

Register State	Short Form	Register Type	Address	Initial State
DCO control register	DCOCTL	Read/write	056h	060h
Basic clock system control 1	BCSCCTL1	Read/write	057h	084h
Basic clock system control 2	BCSCCTL2	Read/write	058h	reset

Basic Clock Systems-control registers(detail)

➤ Digitally-Controlled Oscillator (DCO) Clock-Frequency Control

DCOCTL is loaded with a value of 060h with a valid PUC condition.

0	7								
DCOCTL	DCO.2	DCO.1	DCO.0	MOD.4	MOD.3	MOD.2	MOD.1	MOD.0	
056H	0	1	1	0	0	0	0	0	

MOD.0 .. MOD.4: The MOD constant defines how often the discrete frequency f_{DCO+1} is used within a period of 32 DCOCLK cycles.

During the remaining clock cycles (32-MOD) the discrete frequency f_{DCO} is used. When the DCO constant is set to seven, no modulation is possible since the highest feasible frequency has then been selected.

DCO.0 .. DCO.2: The DCO constant defines which one of the eight discrete frequencies is selected. The frequency is defined by the current injected into the dc generator.

Basic Clock Systems-control registers(detail)

➤ Oscillator and Clock Control Register

BCSCTL1 is affected by a valid PUC or POR condition.

	7							0
BCSCTL1	XT2Off	XTS	DIVA.1	DIVA.0	XT5V	Rsel.0	Rsel.1	Rsel.2
057h	1	0	0	0	0	1	0	0

Bit0 to Bit2: The internal resistor is selected in eight different steps.

Rsel.0 to Rsel.2 The value of the resistor defines the nominal frequency.

The lowest nominal frequency is selected by setting Rsel=0.

Bit3, XT5V: XT5V should always be reset.

Bit4 to Bit5: The selected source for ACLK is divided by:

DIVA = 0: 1

DIVA = 1: 2

DIVA = 2: 4

DIVA = 3: 8

Basic Clock Systems-control registers(detail)

Bit6, XTS: The LFXT1 oscillator operates with a low-frequency or with a high-frequency crystal:

XTS = 0: The low-frequency oscillator is selected.

XTS = 1: The high-frequency oscillator is selected.

The oscillator selection must meet the external crystal's operating condition.

Bit7, XT2Off: The XT2 oscillator is switched on or off:

XT2Off = 0: the oscillator is on

XT2Off = 1: the oscillator is off if it is not used for MCLK or SMCLK.

Basic Clock Systems-control registers(detail)

BCSCTL2 is affected by a valid PUC or POR condition.

7 _____ 0

BCSCTL2 SELM.1 SELM.0 DIVM.1 DIVM.0 SELS DIVS.1 DIVS.0 DCOR
058h

Bit0, DCOR: The DCOR bit selects the resistor for injecting current into the dc generator. Based on this current, the oscillator operates if activated.

DCOR = 0: Internal resistor on, the oscillator can operate. The fail-safe mode is on.

DCOR = 1: Internal resistor off, the current must be injected externally if the DCO output drives any clock using the DCOCLK.

Bit1, Bit2: The selected source for SMCLK is divided by:

DIVS.1 .. DIVS.0 DIVS = 0:1

DIVS = 1: 2

DIVS = 2: 4

DIVS = 3: 8

Basic Clock Systems-control registers(detail)

Bit3, SELS: Selects the source for generating SMCLK:

SELS = 0: Use the DCOCLK

SELS = 1: Use the XT2CLK signal (in three-oscillator systems)

or

LFXT1CLK signal (in two-oscillator systems)

Bit4, Bit5: The selected source for MCLK is divided by DIVM.0 .. DIVM.1

DIVM = 0: 1

DIVM = 1: 2

DIVM = 2: 4

DIVM = 3: 8

Bit6, Bit7: Selects the source for generating MCLK:

SELM.0 .. SELM.1

SELM = 0: Use the DCOCLK

SELM = 1: Use the DCOCLK

SELM = 2: Use the XT2CLK (x13x and x14x devices)

or

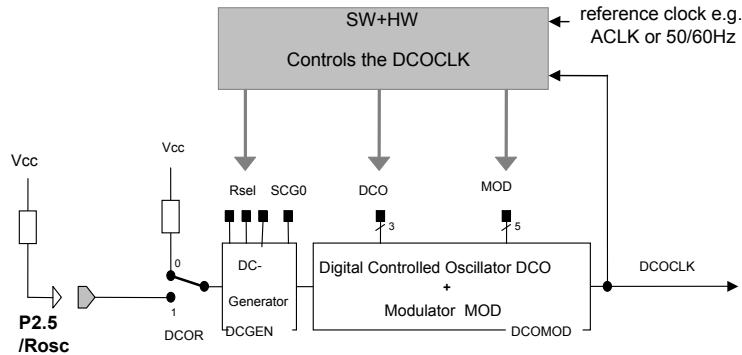
Use the LFXT1CLK (x11x(1) devices)

SELM = 3: Use the LFXT1CLK

Basic Clock Systems-software FII idea

➤ Basic Clock DCO is an open loop - close with SW+HW

- ❖ A reference frequency e.g. ACLK or 50/60Hz can be used to measure DCOCLK's
- ❖ **Initialization or Periodic** software set and stabilizes DCOCLK over reference clock
- ❖ DCOCLK is programmable 100kHz - 5Mhz and stable over voltage and temperature



CPE/EE 421/521 Microcomputers

27

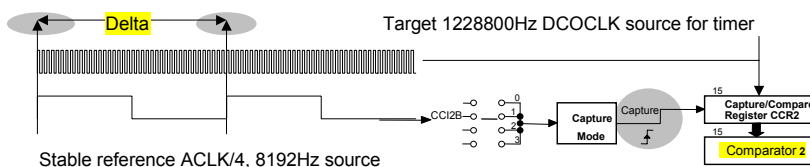
Basic Clock Systems-software FII implementation

➤ Example: Set DCOCLK= 1228800, ACLK= 32768

- ❖ ACLK/4 captured on CCI2B, DCOCLK is clock source for Timer_A
- ❖ Comparator2 HW captures SMCLK (1228800Hz) in one ACLK/4 (8192Hz) period
- ❖ Target Delta = $1228800/8192 = 150$

```

CCI2BInt ... ; Compute Delta
    cmp    #150,Delta ; Delta= 1228800/8192
    jlo   IncDCO    ; JMP to IncDCO
DecDCO  dec    &DCOCTL ; Decrease DCOCLK
    reti
IncDCO  inc    &DCOCTL ; Increase DCOCLK
    reti
    
```



CPE/EE 421/521 Microcomputers

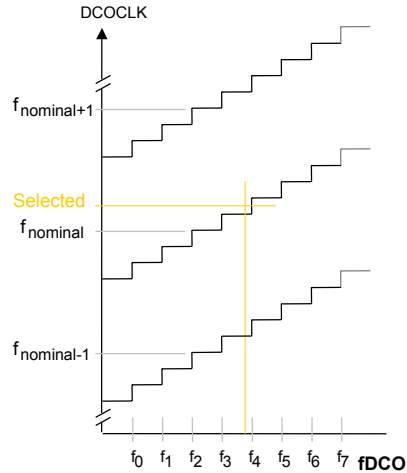
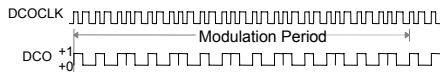
28

Basic Clock Systems-DCO TAPS

➤ **DCOCLK frequency control**

- ❖ nominal - injected current into DC generator
 - 1) internal resistors Rsel2, Rsel1 and Rsel0
 - 2) an external resistor at Rsc (P2.5/11x)
- ❖ Control bits DCO0 to DCO2 set fDCO tap
- ❖ Modulation bits MOD0 to MOD4 allow mixing of fDCO and fDCO+1 for precise frequency generation

Example	Frequency	Cycle time
Selected:	1000kHz	1000 nsec
f3:	943kHz	1060 nsec
f4:	1042kHz	960 nsec
MOD=19		

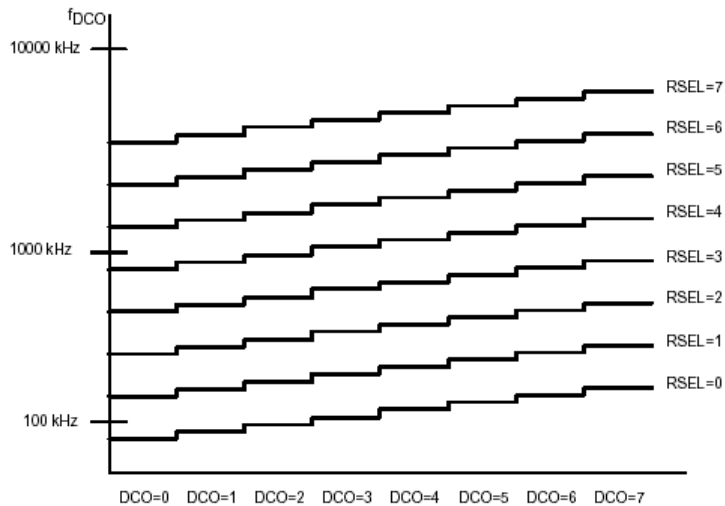


Cycle_time = ((32-MOD)*t1+MOD*t2)/32 = 1000.625 ns, selected frequency ≈ 1 MHz.

F149 default DCO clock setting

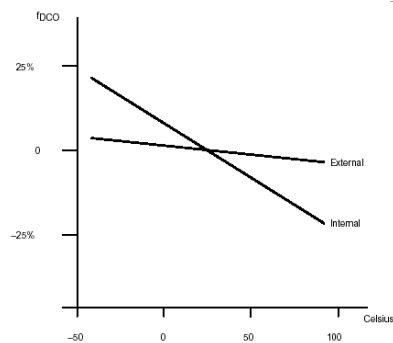
PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT	
f(DCO03)	Rsel = 0, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	0.08	0.12	0.15	MHz
		VCC = 3 V	0.08	0.13	0.16	
f(DCO13)	Rsel = 1, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	0.14	0.19	0.23	MHz
		VCC = 3 V	0.14	0.18	0.22	
f(DCO23)	Rsel = 2, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	0.22	0.30	0.36	MHz
		VCC = 3 V	0.22	0.28	0.34	
f(DCO33)	Rsel = 3, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	0.37	0.49	0.59	MHz
		VCC = 3 V	0.37	0.47	0.56	
f(DCO43)	Rsel = 4, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	0.61	0.77	0.93	MHz
		VCC = 3 V	0.61	0.75	0.90	
f(DCO53)	Rsel = 5, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	1	1.2	1.5	MHz
		VCC = 3 V	1	1.3	1.5	
f(DCO63)	Rsel = 6, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	1.6	1.9	2.2	MHz
		VCC = 3 V	1.69	2.0	2.29	
f(DCO73)	Rsel = 7, DCO = 3, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	2.4	2.9	3.4	MHz
		VCC = 3 V	2.7	3.2	3.65	
f(DCO47)	Rsel = 4, DCO = 7, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V/3 V	fDCO40 × 1.7	fDCO40 × 2.1	fDCO40 × 2.5	MHz
f(DCO77)	Rsel = 7, DCO = 7, MOD = 0, DCOR = 0, TA = 25°C	VCC = 2.2 V	4	4.5	4.9	MHz
		VCC = 3 V	4.4	4.9	5.4	
SR(Rsel)	SR = fRsel+1 / fRsel	VCC = 2.2 V/3 V	1.35	1.65	2	
SDCO	SDCO = fDCO+1 / fDCO	VCC = 2.2 V/3 V	1.07	1.12	1.16	
Dt	Temperature drift, Rsel = 4, DCO = 3, MOD = 0 (see Note 30)	VCC = 2.2 V	-0.31	-0.36	-0.40	%/°C
		VCC = 3 V	-0.33	-0.38	-0.43	
DV	Drift with VCC variation, Rsel = 4, DCO = 3, MOD = 0 (see Note 30)	VCC = 2.2 V/3 V	0	5	10	%V

Range (RSELx) and Steps (DCOx)



External Resistor

- The DCO temperature coefficient can be reduced by using an external resistor ROSC to source the current for the DC generator.
- ROSC also allows the DCO to operate at higher frequencies.
 - ❖ Internal resistor nominal value is approximately 200 kOhm => DCO to operate up to 5 MHz.
 - ❖ External ROSC of approximately 100 kOhm => the DCO can operate up to approximately 10 MHz.



DCO Modulator

- To produce 1 an intermediate effective frequency between f_{DCO} and $f_{\text{DCO}+1}$

The modulator mixing formula is:

$$t = (32 - \text{MODx}) \times t_{\text{DCO}} + \text{MODx} \times t_{\text{DCO}+1}$$

Fail Safe Operation

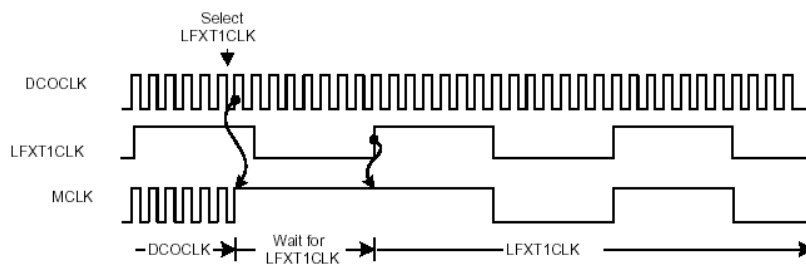
- Basic module incorporates an oscillator-fault detection fail-safe feature.
- The oscillator fault detector is an analog circuit that monitors the LFXT1CLK (in HF mode) and the XT2CLK.
- An oscillator fault is detected when either clock signal is not present for approximately 50 μs .
 - ❖ When an oscillator fault is detected, and when MCLK is sourced from either LFXT1 in HF mode or XT2, MCLK is automatically switched to the DCO for its clock source.

Fail Safe Operation

- When OFIFG is set and OFIE is set, an NMI interrupt is requested. The NMI interrupt service routine can test the OFIFG flag to determine if an oscillator fault occurred. The OFIFG flag must be cleared by software.

Synchronization of clock signals

- To avoid race conditions
 - ❖ The current clock cycle continues until the next rising edge.
 - ❖ The clock remains high until the next rising edge of the new clock.
 - ❖ The new clock source is selected and continues with a full high period.



Basic Clock Systems-Examples

➤ How to select the Crystal Clock

```
void selectclock(void)
{
    IFG2=0;          /* reset interrupt flag register 1 */
    IFG1=0;          /* reset interrupt flag register 2 */
    BCSTL1|=XTS;    /*attach HF crystal (4MHz) to XIN/XOUT */
    do {
        /*wait in loop until crystal is stable*/
        IFG1&=~OFIFG;
    }while(OFIFG&IFG1);

    Delay();
    IFG1&=~OFIFG;    /*Reset osc. fault flag again*/
}
```

➤ How to select a clock for MCLK

```
BCSTL2=SELM0+SELM1;    /*Then set MCLK same as LFXT1CLK*/
TACTL=TASSEL0+TACLR+ID1; /*USE ACLK/4 AS TIMER_A INPUT CLOCK
                          (1MHz) */
```

Basic Clock Systems-Examples

➤ Adjusting the Basic Clock

The control registers of the Basic Clock are under full software control. If clock requirements other than those of the default from PUC are necessary, the Basic Clock can be configured or reconfigured by software at any time during program execution.

- ACLKGEN from LFXT1 crystal, resonator, or external-clock source and divided by 1, 2, 4, or 8. If no LFXTCLK clock signal is needed in the application, the OscOff bit should be set in the status register.
- SCLKGEN from LFXTCLK, DCOCLK, or XT2CLK (x13x and x14x only) and divided by 1, 2, 4, or 8. The SCG1 bit in the status register enables or disables SMCLK.
- MCLKGEN from LFXTCLK, DCOCLK, or XT2CLK (x13x and x14x only) and divided by 1, 2, 4, or 8. When set, the CPUOff bit in the status register enables or disables MCLK.
- DCOCLK frequency is adjusted using the RSEL, DCO, and MOD bits. The DCOCLK clock source is stopped when not used, and the dc generator can be disabled by the SCG0 bit in the status register (when set).
- The XT2 oscillator sources XT2CLK (x13x and x14x only) by clearing the XT2Off bit.

Interrupt Service Routines

➤ Interrupt Service Routine declaration

```
// Func. declaration
Interrupt[int_vector] void myISR (Void);

Interrupt[int_vector] void myISR (Void)
{
// ISR code
}
```

❑ EXAMPLE

```
Interrupt[TIMERA0_VECTOR] void myISR (Void);

Interrupt[TIMERA0_VECTOR] void myISR (Void)
{
// ISR code
}
```

Interrupt Service Routines

➤ MSP430 interrupt vectors (int_vector)

```
➤ /*****
➤ * Interrupt Vectors (offset from 0xFFE0)
➤ *****/

➤ #define PORT2_VECTOR      1 * 2 /* 0xFFE2 Port 2 */
➤ #define UART1TX_VECTOR   2 * 2 /* 0xFFE4 UART 1 Transmit */
➤ #define UART1RX_VECTOR   3 * 2 /* 0xFFE6 UART 1 Receive */
➤ #define PORT1_VECTOR     4 * 2 /* 0xFFE8 Port 1 */
➤ #define TIMERA1_VECTOR   5 * 2 /* 0xFFEA Timer A CC1-2, TA */
➤ #define TIMERA0_VECTOR   6 * 2 /* 0xFFEC Timer A CC0 */
➤ #define ADC_VECTOR       7 * 2 /* 0xFFEE ADC */
➤ #define UART0TX_VECTOR   8 * 2 /* 0xFFFF0 UART 0 Transmit */
➤ #define UART0RX_VECTOR   9 * 2 /* 0xFFFF2 UART 0 Receive */
➤ #define WDT_VECTOR      10 * 2 /* 0xFFFF4 Watchdog Timer */
➤ #define COMPARATORA_VECTOR 11 * 2 /* 0xFFFF6 Comparator A */
➤ #define TIMERB1_VECTOR   12 * 2 /* 0xFFFF8 Timer B 1-7 */
➤ #define TIMERB0_VECTOR   13 * 2 /* 0xFFFFA Timer B 0 */
➤ #define NMI_VECTOR       14 * 2 /* 0xFFFFC Non-maskable */
➤ #define RESET_VECTOR     15 * 2 /* 0xFFFFE Reset [Highest Pr.] */
```

Watchdog Timer-General

General

The primary function of the watchdog-timer module (WDT) 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 work as an interval timer, to generate an interrupt after the selected time interval.

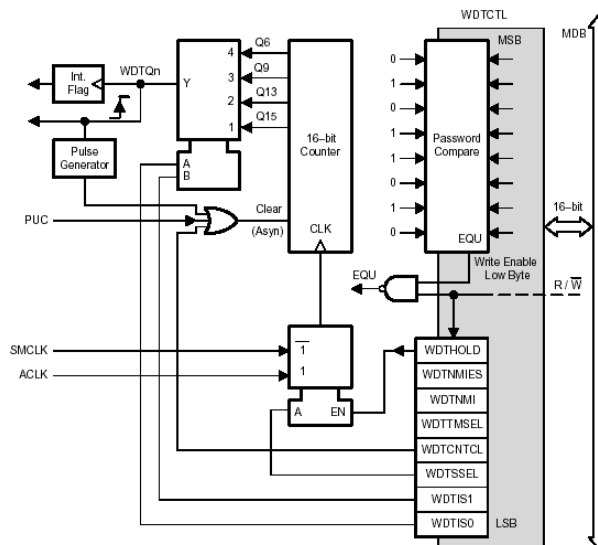
Features of the Watchdog Timer include:

- Eight software-selectable time intervals
- Two operating modes: as watchdog or interval timer
- Expiration of the time interval in watchdog mode, which generates a system reset; or in timer mode, which generates an interrupt request
- Safeguards which ensure that writing to the WDT control register is only possible using a password
- Support of ultralow-power using the hold mode

Watchdog/Timer two functions:

- SW Watchdog Mode
- Interval Timer Mode

Watchdog Timer-Diagram

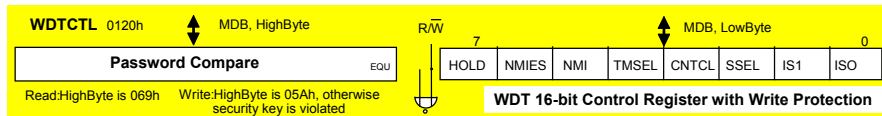


Watchdog Timer-Registers

❑ Watchdog Timer Counter

The watchdog-timer counter (WDCNT) is a 16-bit up-counter that is not directly accessible by software. The WDCNT is controlled through the watchdog-timer control register (WDTCTL), which is a 16-bit read/write register located at the low byte of word address 0120h. Any read or write access must be done using word instructions with no suffix or .w suffix. In both operating modes (watchdog or timer), it is only possible to write to WDTCTL using the correct password.

❑ Watchdog Timer Control Register



Bits 0, 1: Bits ISO and IS1 select one of four taps from the WDCNT, as described in following table. Assuming $f_{\text{crystal}} = 32,768 \text{ Hz}$ and $f_{\text{System}} = 1 \text{ MHz}$, the following intervals are possible:

Watchdog Timer-Registers

SSEL	IS1	ISO	Interval [ms]		Table: WDCNT Taps
0	1	1	0.064	$t_{\text{SMCLK}} \times 2^6$	
0	1	0	0.5	$t_{\text{SMCLK}} \times 2^9$	
1	1	1	1.9	$t_{\text{ACLK}} \times 2^6$	
0	0	1	8	$t_{\text{SMCLK}} \times 2^{13}$	
1	1	0	16.0	$t_{\text{ACLK}} \times 2^9$	
0 (reset)	0	0	32	$t_{\text{SMCLK}} \times 2^{15}$ <- Value after PUC	
1	0	1	250	$t_{\text{ACLK}} \times 2^{13}$	
1	0	0	1000	$t_{\text{ACLK}} \times 2^{15}$	

Bit 2: The SSEL bit selects the clock source for WDCNT.

SSEL = 0: WDCNT is clocked by SMCLK .

SSEL = 1: WDCNT is clocked by ACLK.

Bit 3: Counter clear bit. In both operating modes, writing a 1 to this bit restarts the WDCNT at 00000h. The value read is not defined.

Bit 4: The TMSSEL bit selects the operating mode: watchdog or timer.

TMSSEL = 0: Watchdog mode

TMSSEL = 1: Interval-timer mode

Bit 5: The NMI bit selects the function of the RST/NMI input pin. It is

Watchdog Timer-Registers

NMI = 0: The RST/NMI input works as reset input.

As long as the RST/NMI pin is held low, the internal signal is active (level sensitive).

NMI = 1: The RST/NMI input works as an edge-sensitive non-maskable interrupt input.

Bit 6: If the NMI function is selected, this bit selects the activating edge of the RST/NMI input. It is cleared by the PUC signal.

NMIES = 0: A rising edge triggers an NMI interrupt.

NMIES = 1: A falling edge triggers an NMI interrupt.

CAUTION: Changing the NMIES bit with software can generate an NMI interrupt.

Bit 7: This bit stops the operation of the watchdog counter. The clock multiplexer is disabled and the counter stops incrementing. It holds the last value until the hold bit is reset and the operation continues. It is cleared by the PUC signal.

HOLD = 0: The WDT is fully active.

HOLD = 1: The clock multiplexer and counter are stopped.

Watchdog Timer-Interrupt Function

- ❑ **The Watchdog Timer (WDT) uses two bits in the SFRs for interrupt control.**

The WDT interrupt flag (WDTIFG) (located in IFG1.0, initial state is reset)

The WDT interrupt enable (WDTIE) (located in IE1.0, initial state is reset)

- ❖ When using the watchdog mode, the WDTIFG flag is used by the reset interrupt service routine to determine if the watchdog caused the device to reset. If the flag is set, then the Watchdog Timer initiated the reset condition (either by timing out or by a security key violation). If the flag is cleared, then the PUC was caused by a different source. See chapter 3 for more details on the PUC and POR signals.
- ❖ When using the Watchdog Timer in interval-timer mode, the WDTIFG flag is set after the selected time interval and a watchdog interval-timer interrupt is requested. The interrupt vector address in interval-timer mode is different from that in watchdog mode. In interval-timer mode, the WDTIFG flag is reset automatically when the interrupt is serviced.
- ❖ The WDTIE bit is used to enable or disable the interrupt from the Watchdog Timer when it is being used in interval-timer mode. Also, the GIE bit enables or disables the interrupt from the Watchdog Timer when it is being used in interval-timer mode.

Watchdog Timer-Timer Mode

- ❖ Setting WDTCTL register bit TMSEL to 1 selects the timer mode. This mode provides periodic interrupts at the selected time interval. A time interval can also be initiated by writing a 1 to bit CNTCL in the WDTCTL register.
- ❖ When the WDT is configured to operate in timer mode, the WDTIFG flag is set after the selected time interval, and it requests a standard interrupt service. The WDT interrupt flag is a single-source interrupt flag and is automatically reset when it is serviced. The enable bit remains unchanged. In interval-timer mode, the WDT interrupt-enable bit and the GIE bit must be set to allow the WDT to request an interrupt. The interrupt vector address in timer mode is different from that in watchdog mode.

Watchdog Timer-Examples

❑ How to select timer mode

```
/* WDT is clocked by fACLK (assumed 32Khz) */
WDTCL=WDT_ADLY_250; // WDT 250MS/4 INTERVAL TIMER
IE1 |=WDTIE;        // ENABLE WDT INTERRUPT
```

❑ How to stop watchdog timer

```
WDTCTL=WDTFW + WDT HOLD ; // stop watchdog timer
```

❑ Assembly programming

```
WDT_key      .equ    05A00h                ; Key to access WDT
WDTStop      mov     #(WDT_Key+80h),&WDTCTL ; Hold Watchdog
WDT250       mov     #(WDT_Key+1Dh),&WDTCTL ; WDT, 250ms Interval
```