MSP430x1xx Microcontrollers
Basic Clock Module

CPE 621 Advanced Microcomputer Techniques
Dr. Emil Jovanov

Basic Clock Systems

- MSP430 Clock System
  - Low System Cost
  - Low Power

- Variety of Operating Modes driven by application, software selectable
- Support for Burst Mode that when activated system starts and reacts rapidly
- Stability over Voltage and Temperature
Basic Clock Systems

Basic Clock System-MSP430x1xxx

- One DCO, digital controlled oscillator
  - Generated on-chip
  - RC-type frequency controller by SW + HW

- One LF/XT oscillator
  - LF: 32768Hz
  - XT: 450kHz ... 8MHz

- Second LF/XT2 oscillator
  - Future Expansion

- Clocks:
  - ACLK auxiliary clock
  - MCLK main system clock
  - SMCLK sub main system clock

DCOCLK Generated on-chip with 6μS 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**

The DCO-Generator is connected to pin P2.5/Rosc if DCOR control bit is set.
The port pin P2.5/Rosc is selected if DCOR control bit is reset (initial state).

---

**Basic Clock Systems-control registers**

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

- **BCSCTL2** 056h

- **BCSCTL1** 057h

- **DCOCTL** 058h

  - **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, BCSCTL1, and BCSCTL2, 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.

<table>
<thead>
<tr>
<th>Register</th>
<th>Short Form</th>
<th>Register Type</th>
<th>Address</th>
<th>Initial State</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCO control register</td>
<td>DCOCTL</td>
<td>Read/write</td>
<td>056h</td>
<td>060h</td>
</tr>
<tr>
<td>Basic clock system control 1</td>
<td>BCSCTL1</td>
<td>Read/write</td>
<td>057h</td>
<td>084h</td>
</tr>
<tr>
<td>Basic clock system control 2</td>
<td>BCSCTL2</td>
<td>Read/write</td>
<td>058h</td>
<td>reset</td>
</tr>
</tbody>
</table>

Digitally-Controlled Oscillator (DCO) Clock-Frequency Control

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

<table>
<thead>
<tr>
<th>DCOCTL</th>
<th>DCO.2</th>
<th>DCO.1</th>
<th>DCO.0</th>
<th>MOD.4</th>
<th>MOD.3</th>
<th>MOD.2</th>
<th>MOD.1</th>
<th>MOD.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>056h</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

MOD.0 .. MOD.4: The MOD constant defines how often the discrete frequency $\frac{f_{DCO}}{32}$ 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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>XT2Off</td>
</tr>
<tr>
<td>6</td>
<td>XTS</td>
</tr>
<tr>
<td>5</td>
<td>DIVA.1</td>
</tr>
<tr>
<td>4</td>
<td>DIVA.0</td>
</tr>
<tr>
<td>3</td>
<td>XT5V</td>
</tr>
<tr>
<td>2</td>
<td>Rsel.0</td>
</tr>
<tr>
<td>1</td>
<td>Rsel.1</td>
</tr>
<tr>
<td>0</td>
<td>Rsel.2</td>
</tr>
</tbody>
</table>

  **BCSCTL1**: 057h

  - Bit 0 to Bit 2: 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.
  - Bit 3, XT5V: XT5V should always be reset.
  - Bit 4 to Bit 5: The selected source for ACLK is divided by:
    - `DIVA = 0`: 1
    - `DIVA = 1`: 2
    - `DIVA = 2`: 4
    - `DIVA = 3`: 8
  - Bit 6, 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.

  - Bit 7, 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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SELM.1</td>
</tr>
<tr>
<td>6</td>
<td>SELM.0</td>
</tr>
<tr>
<td>5</td>
<td>DIVM.1</td>
</tr>
<tr>
<td>4</td>
<td>DIVM.0</td>
</tr>
<tr>
<td>3</td>
<td>SELS</td>
</tr>
<tr>
<td>2</td>
<td>DIVS.1</td>
</tr>
<tr>
<td>1</td>
<td>DIVS.0</td>
</tr>
<tr>
<td>0</td>
<td>DCOR</td>
</tr>
</tbody>
</table>

  **BCSCTL2**: 058h

  - Bit 0, 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.

  - Bit 1, Bit 2: 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 FI1 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
Basic Clock Systems-software FII implementation

Example: Set DCOCLK= 1228800, ACLK= 32768
1. ACLK/4 captured on CCI2B, DCOCLK is clock source for Timer_A
2. Comparator2 HW captures SMCLK (1228800Hz) in one ACLK/4 (8192Hz) period
3. Target Delta = 1228800/8192 = 150

CCI2BInt ...
    cmp #150,Delta
    jlo IncDCO
    DecDCO dec &DCOCTL
    reti
IncDCO inc &DCOCTL
    reti

Target 1228800Hz DCOCLK source for timer
Stable reference ACLK/4, 8192Hz source

Basic Clock Systems-DCO TAPS

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

Example
Selected: Frequency  Cycle time
f0: 1000kHz  1000 nsec
f1: 943kHz 1060 nsec
f2: 1042kHz 960 nsec
MOD=19

Cycle_time = ((32-MOD)*t1+MOD*t2)/32 = 1000.625 ns, selected frequency ≈ 1 MHz.
MSP430x1xx Microcontrollers
DCO Clock Example

CPE 621 Advanced Microcomputer Techniques
Dr. Emil Jovanov

### F149 default DCO clock setting

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rref = 0, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.08</td>
<td>0.12</td>
<td>0.15</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0.08</td>
<td>0.13</td>
<td>0.16</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 1, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.14</td>
<td>0.19</td>
<td>0.23</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0.14</td>
<td>0.16</td>
<td>0.22</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 2, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.22</td>
<td>0.28</td>
<td>0.32</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0.22</td>
<td>0.26</td>
<td>0.34</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 3, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.37</td>
<td>0.49</td>
<td>0.59</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0.37</td>
<td>0.47</td>
<td>0.56</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 4, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.61</td>
<td>0.77</td>
<td>0.93</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0.61</td>
<td>0.75</td>
<td>0.90</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 5, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>1.0</td>
<td>1.3</td>
<td>1.5</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>1.0</td>
<td>1.3</td>
<td>1.5</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 6, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>1.6</td>
<td>1.9</td>
<td>2.2</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>1.6</td>
<td>2.0</td>
<td>2.3</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 7, DCO = 3, MOD = 6, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>2.4</td>
<td>2.9</td>
<td>3.4</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>2.7</td>
<td>3.2</td>
<td>3.6</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 4, DCO = 7, MOD = 0, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>1.9 V5</td>
<td>4.5 V5</td>
<td>6.5 V5</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>4.4 V5</td>
<td>4.9 V5</td>
<td>5.4 V5</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 7, DCO = 7, MOD = 0, OCR = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>1.35</td>
<td>1.85</td>
<td>2</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>4.4</td>
<td>4.9</td>
<td>5.4</td>
<td>MHz</td>
</tr>
<tr>
<td>Rref = 4, DCO = 3, MOD = 0, TA = 25°C</td>
<td>VCC = 2.2 V</td>
<td>0.7</td>
<td>1.12</td>
<td>1.16</td>
<td>K/°C</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>-0.1</td>
<td>0.36</td>
<td>0.43</td>
<td>K/°C</td>
</tr>
<tr>
<td>Drift with VCC variation, Rref = 4, DCO = 3, MOD = 0</td>
<td>VCC = 2.2 V</td>
<td>-0.1</td>
<td>0.38</td>
<td>0.43</td>
<td>%/V</td>
</tr>
<tr>
<td></td>
<td>VCC = 3 V</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>%/V</td>
</tr>
</tbody>
</table>
Basic Clock Systems-DCO TAPS

- **DCOCLK frequency control**
  
  1. Nominal - injected current into DC generator
  2. Internal resistors Rsel2, Rsel1, and Rsel0
  3. An external resistor at Rosc (P2.5/11x)

- Control bits DCO0 to DCO2 set DCO tap

- Modulation bits MOD0 to MOD4 allow mixing of fDCO and fDCO+1 for precise frequency generation

```
Example  
Selected:  
<table>
<thead>
<tr>
<th>Frequency</th>
<th>Cycle time</th>
</tr>
</thead>
<tbody>
<tr>
<td>f3: 1kHz</td>
<td>1000 nsec</td>
</tr>
<tr>
<td>f4: 943kHz</td>
<td>1060 nsec</td>
</tr>
<tr>
<td>MOD=19</td>
<td></td>
</tr>
</tbody>
</table>
```

**Cycle time** = ((32-MOD)×t1+MOD×t2)/32 = 1000.625 ns, selected frequency = 1 MHz.
F149 DCO clock example

F149 MOD example
Mod:0,2,6
Basic Clock Systems-Examples

How to select the Crystal Clock

```c
void selectclock(void)
{
    IFG2=0;   /* reset interrupt flag register 1 */
    IFG1=0;   /* reset interrupt flag register 2 */
    BCSCTL1|=XTS; /* attach HF crystal (4MHz) to XIN/XOUT */
    do {
        IFG1&=~OFIFG;  /* wait in loop until crystal is stable*/
        } while(OFIFG&IFG1);
    Delay();
    IFG1&=~OFIFG;  /* reset osc. fault flag again */
}
```

How to select a clock for MCLK

```c
BCSCTL2=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**
  ```c
  // Func. declaration
  Interrupt[int_vector] void myISR (Void);
  Interrupt[int_vector] void myISR (Void)
  {
    // ISR code
  }
  ```

- **EXAMPLE**
  ```c
  Interrupt[TIMERA0_VECTOR] void myISR (Void);
  Interrupt[TIMERA0_VECTOR] void myISR (Void)
  {
    // ISR code
  }
  ```

- **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  /* 0xFFF0 UART 0 Transmit */
  #define UART0RX_VECTOR      9 * 2  /* 0xFFF2 UART 0 Receive */
  #define WDT_VECTOR          10 * 2 /* 0xFFF4 Watchdog Timer */
  #define COMPARATORA_VECTOR  11 * 2 /* 0xFFF6 Comparator A */
  #define TIMERB1_VECTOR      12 * 2 /* 0xFFF8 Timer B 1-7 */
  #define TIMERB0_VECTOR      13 * 2 /* 0xFFF8 Timer B 0 */
  #define NMI_VECTOR          14 * 2 /* 0xFFF8 Non-maskable */
  #define RESET_VECTOR        15 * 2 /* 0xFFF8 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-Diagram

Watchdog Timer-Registers

Watchdog Timer Counter
The watchdog-timer counter (WDTCNT) is a 16-bit up-counter that is not directly accessible by software. The WDTCNT 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 IS0 and IS1 select one of four taps from the WDTCNT, as described in following table. Assuming \( f_{\text{crystal}} = 32,768 \) Hz and \( f_{\text{System}} = 1 \) MHz, the following intervals are possible:

Table: WDTCNT Taps

CPE 621 MSP430X14X Microcontroller 26
**Watchdog Timer-Registers**

<table>
<thead>
<tr>
<th>SSEL</th>
<th>IS1</th>
<th>IS0</th>
<th>Interval [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.064</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.9</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>16.0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>250</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1000</td>
</tr>
</tbody>
</table>

Bit 2: The SSEL bit selects the clock source for WDTCNT.
- SSEL = 0: WDTCNT is clocked by SMCLK.
- SSEL = 1: WDTCNT is clocked by ACLK.

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

Bit 4: The TMSEL bit selects the operating mode: watchdog or timer.
- TMSEL = 0: Watchdog mode
- TMSEL = 1: Interval-timer mode

Bit 5: The NMI bit selects the function of the RST/NMI input pin. It is cleared by the PUC signal.
- 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.
- 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
  ```c
  /* 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
  ```c
  WDTCTL=WDTM + WDTHOLD ; // stop watchdog timer
  ```

- Assembly programming
  ```assembly
  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
  ```