Real Time Interrupt (RTI)

The RTI can be used to generate a periodic hardware interrupt. We have not discussed the interrupts much yet and we will cover it in more detail soon. For now, we will just like to use this feature to generate an accurate interrupt that happens every 1[ms] independently from what happens in the main loop. The other convenient aspect of the RTI is that it is gated by the **OSCCLK**, which is our **16[MHZ]** crystal, therefore it does not get affected if we increase the Bus Speed using the PLL.

A. Configuration

There are only two registers to configure to get the RTI setup properly:

2.3.2.8 CRG RTI Control Register (RTICTL)

RTDEC	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
1	0	0	1	0	1	1	1

Table 2-6. RTICTL Field Descriptions

Field	Description
7 RTDEC	**Decimal or Binary Divider Select Bit** — RTDEC selects decimal or binary based prescaler values. 0 Binary based divider value. See Table 2-7 1 Decimal based divider value. See Table 2-8
6–4 RTR[6:4]	**Real Time Interrupt Prescale Rate Select Bits** — These bits select the prescale rate for the RTI. See Table 2-7 and Table 2-8.
3–0 RTR[3:0]	**Real Time Interrupt Modulus Counter Select Bits** — These bits select the modulus counter target value to provide additional granularity.Table 2-7 and Table 2-8 show all possible divide values selectable by the RTICTL register. The source clock for the RTI is OSCCLK.

- We **SET(1) RTDEC** to select Decimal prescaler
- According to the prescale rate (RTR), we configure **BIT6 - BIT4 to 001_b** = 2x10^3, so we can divide the 16MHz Clock (Crystal) by 2,000. Therefore, 16MHz / 2,000 = 8KHz
- According to the Modulus counter setting, we configure **BIT3 - BIT0 to 0111_b** (Mod8 Counter) so we can divide even futher the now 8KHz clock by 8 so we can get a perfect 1KHz clock. Therefore, 8KHz / 8 = 1KHz (1ms event).
-

```
-       Decimal, divider 2000, mod8 counter -> 1ms
-       RTICTL = 0b10010111;
```

RTR[3:0]	RTR[6:4] =							
	000 ($1x10^3$)	001 ($2x10^3$)	010 ($5x10^3$)	011 ($10x10^3$)	100 ($20x10^3$)	101 ($50x10^3$)	110 ($100x10^3$)	111 ($200x10^3$)
0000 (÷1)	$1x10^3$	$2x10^3$	$5x10^3$	$10x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$
0001 (÷2)	$2x10^3$	$4x10^3$	$10x10^3$	$20x10^3$	$40x10^3$	$100x10^3$	$200x10^3$	$400x10^3$
0010 (÷3)	$3x10^3$	$6x10^3$	$15x10^3$	$30x10^3$	$60x10^3$	$150x10^3$	$300x10^3$	$600x10^3$
0011 (÷4)	$4x10^3$	$8x10^3$	$20x10^3$	$40x10^3$	$80x10^3$	$200x10^3$	$400x10^3$	$800x10^3$
0100 (÷5)	$5x10^3$	$10x10^3$	$25x10^3$	$50x10^3$	$100x10^3$	$250x10^3$	$500x10^3$	$1x10^6$
0101 (÷6)	$6x10^3$	$12x10^3$	$30x10^3$	$60x10^3$	$120x10^3$	$300x10^3$	$600x10^3$	$1.2x10^6$
0110 (÷7)	$7x10^3$	$14x10^3$	$35x10^3$	$70x10^3$	$140x10^3$	$350x10^3$	$700x10^3$	$1.4x10^6$
0111 (÷8)	$8x10^3$	$16x10^3$	$40x10^3$	$80x10^3$	$160x10^3$	$400x10^3$	$800x10^3$	$1.6x10^6$
1000 (÷9)	$9x10^3$	$18x10^3$	$45x10^3$	$90x10^3$	$180x10^3$	$450x10^3$	$900x10^3$	$1.8x10^6$
1001 (÷10)	$10 x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$	$500x10^3$	$1x10^6$	$2x10^6$
1010 (÷11)	$11 x10^3$	$22x10^3$	$55x10^3$	$110x10^3$	$220x10^3$	$550x10^3$	$1.1x10^6$	$2.2x10^6$
1011 (÷12)	$12x10^3$	$24x10^3$	$60x10^3$	$120x10^3$	$240x10^3$	$600x10^3$	$1.2x10^6$	$2.4x10^6$
1100 (÷13)	$13x10^3$	$26x10^3$	$65x10^3$	$130x10^3$	$260x10^3$	$650x10^3$	$1.3x10^6$	$2.6x10^6$
1101 (÷14)	$14x10^3$	$28x10^3$	$70x10^3$	$140x10^3$	$280x10^3$	$700x10^3$	$1.4x10^6$	$2.8x10^6$
1110 (÷15)	$15x10^3$	$30x10^3$	$75x10^3$	$150x10^3$	$300x10^3$	$750x10^3$	$1.5x10^6$	$3x10^6$
1111 (÷16)	$16x10^3$	$32x10^3$	$80x10^3$	$160x10^3$	$320x10^3$	$800x10^3$	$1.6x10^6$	$3.2x10^6$

2.3.2.5 CRG Interrupt Enable Register (CRGINT)

	7	6	5	4	3	2	1	0
R	RTIE	ILAF	0	LOCKIE	0	0	SCMIE	0
W								
Reset	0	1	0	0	0	0	0	0

We just have tpo **SET(1) RTIE** to enable the RTI interrupt

```
CRGINT |= CRGINT_RTIE_MASK;    //0b10000000,      Enable real time Interrupt
```

B. Interrupt Handling

To handle the interrupt we perform the following actions in the Interrupt Service Routine (ISR)

- Clear Flag in the **CRGFLG** register by putting a **(1)** in the **RTIF** bit

	7	6	5	4	3	2	1	0
R	RTIF	PORF	LVRF	LOCKIF	LOCK	TRACK	SCMIF	SCM
W								
Reset	0	1	2	0	0	0	0	0

- We perform any action desired, this action will happen every 1 [ms] in this case, so a good idea is to increment a millisecond counter here.

```
interrupt VectorNumber_Vrti void Vrti_Handler(void)
{
  CRGFLG = CRGFLG_RTIF_MASK;    //clear flag;
  //Perform some action here
}
```