Version 1.1

eRIC_TimerA0:

eRIC has two timers,TimerA0 and TimerA1. TimerA1 is being used in eROS which has only three capture/compare registers. So only TimerA0 is available to use, which has five capture/compare registers. This is also a 16 bit timer.

Refer latest eRIC_eROS_Developers_Manual_x.x for complete list of TimerA0 definitions. Refer SLAU259E document by Texas Instruments to use core TImerA0 registers.

Choosing Clock source, frequency of clock source, period, output Modes, mode of capture/compare and dealing with interrupts or flags is all it needs to be working with TimerA0.

Setting TimerA0:

Timer can be set in one setup or set each feature separately which are explained below.

The Timer count is started as soon as clock source is active and one of the modes is chosen other than Stop mode.

eRIC_TimerA0_Cs(Clocksource); TimerA0 clock source can be set using this.
ClockSource can be:

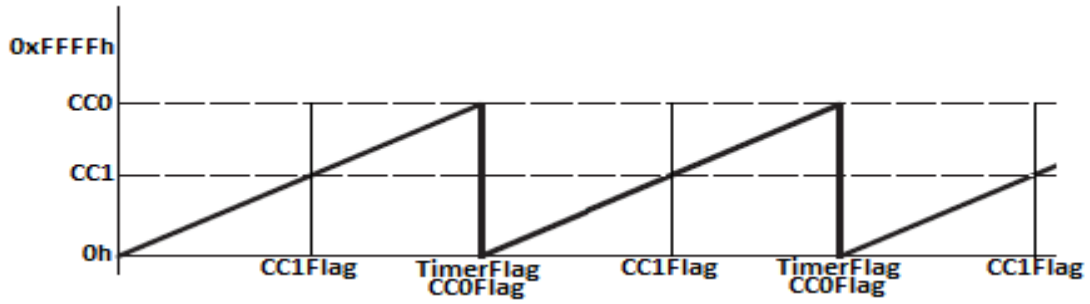
a) eRICTimer_Cs_32k: This can be used to choose Aclk(32768) as clock source for Timer.
b) eRICTimer_Cs_CPU: This can be used to choose SMCLK(upto 20Mhz frequency set by eRIC_SetCpuFrequency()) as clock source for Timer.

eRIC_TimerA0_ClockDivider(Clockdivider); TimerA0 clock can be further divided using this.
Clockdivider can be:

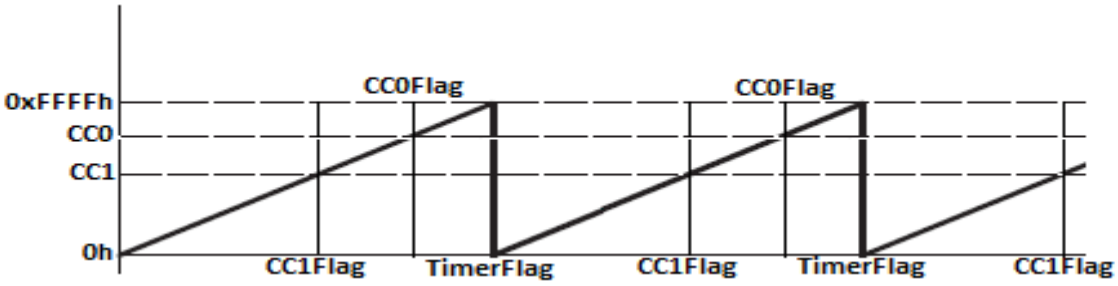
a) eRICTimer_Div_1: The clock source divided by 1
b) eRICTimer_Div_2: The clock source further divided by 2
c) eRICTimer_Div_4: The clock source further divided by 4
d) eRICTimer_Div_8: The clock source further divided by 8

eRIC_TimerA0_Stop(); This completely halts the Timer.

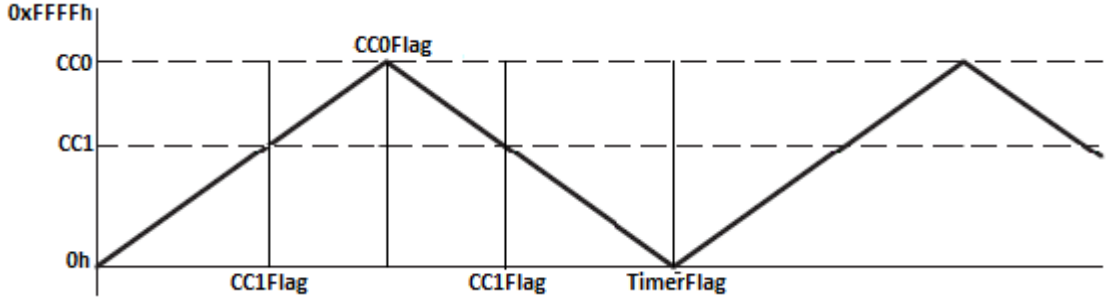
eRIC_TimerA0_UpMode(); Timer repeatedly counts from zero to the value of eRIC_TimerA0_Capture0orCompare0_Data(CC0).
The up mode is used if timer period must be different from 0xFFFFh counts. The timer repeatedly counts up to the value of eRIC_TimerA0_Capture0orCompare0_Data(CC0) which defines the period. If timer value is equal or greater than CC0, the timer immediately restarts counting from zero.
As there are five capture/Compare (CC0,CC1,CC2,CC3,CC4), the corresponding capture/Compare flag is set when timer reaches the value of it. But main timer interrupt flag is set when timer value reaches CC0 as CC0 is the period for this mode. When changing the CC0 while timer is running, if the new period is greater than or equal to the old period or greater than current count value, the timer counts upto new period. Otherwise, the timer rolls to zero.



eRIC_TimerA0_ContinousMode(); Timer repeatedly counts from zero to the value of 0xFFFFh and restarts from zero. CC0 works in the same way as other Capture/Compare registers. The main timer interrupt flag is set when timer counts from 0xFFFFh to zero.



eRIC_TimerA0_UpdownMode(); Updown mode is used if the timer period must be different from 0xFFFFh counts, and if symmetrical pulse generation is needed. The timer repeatedly counts upto the value of CC0 and back down to zero. The period is twice the value in CC0. In this mode CC0 flag and timer flag are set only once during a period. CC0 flag is set when timer counts from CC0-1 to CC0 and timer flag is set when timer completes counting down from 01 to 0h.



eRIC_TimerA0_Reset(); This resets the timer counter to zero , the timer clock divider logic and the count direction.

eRIC_TimerA0_InterruptEnable(); The main timer interrupt is enabled. This is not Capture/Compare interrupt.
eRIC_TimerA0_InterruptDisable();The main timer interrupt is disabled. This is not Capture/Compare interrupt.
eRIC_TimerA0_IsInterruptEnabled(); Returns 1 if interrupt is enabled or 0 if not enabled.

eRIC_TimerA0_InterruptFlag_set(); The main timer interrupt flag is set. This is not Capture/Compare interrupt.
eRIC_TimerA0_InterruptFlag_clear();The main timer interrupt flag is cleared. This is not Capture/Compare interrupt.
eRIC_TimerA0_HasInterrupted(); Returns 1 if interrupt flag is set or 0 if it is cleared.

Alternatively all the above setting can be done in one function as shown below:
eRIC_TimerA0_Setup(CompleteSetup); This can set the timer in one go by passing all features in complete setup.
Completesetup can be sum of combination of below features:

1) ClockSource:
   c) eRICTimer_Cs_32k: This can be used to choose Aclk(32768) as clock source for Timer.
   d) eRICTimer_Cs_CPU: This can be used to choose SMCLK(upto 20Mhz frequency set by eRIC_SetCpuFrequency()) as clock source for Timer.
2) ClockDivider:
   e) eRICTimer_Div_1: The clock source can further be divided by 1
   f) eRICTimer_Div_2: The clock source can further be divided by 2
   g) eRICTimer_Div_4: The clock source can further be divided by 4
   h) eRICTimer_Div_8: The clock source can further be divided by 8
3) Mode:
   a) eRICTimer_stop: This halts the timer
   b) eRICTimer_UpMode: In this mode timer repeatedly counts from zero to the value of CC0.
   c) eRICTimer_ContinousMode: In this mode timer repeatedly counts from zero to 0xFFFFh.
   d) eRICTimer_UpdownMode: In this mode timer repeatedly counts from zero to the value of CC0 and back to zero.
4) TimerReset:
   a) eRICTimer_Reset: This resets the timer counter,clock divider logic and count direction.
5) Interrupt:
   a) eRICTimer_InterruptEnable: This enables the main Timer interrupt not capture/compare interrupt.
   b) eRICTimer_InterruptDisable: This disables the main Timer interrupt not capture/compare interrupt.

   To set Timer with 32k Clock, Divider1,UpMode and enable interrupt, all it needs is this:
   eRIC_TimerA0_Setup(eRICTimer_Cs_32k+ eRICTimer_Div_1+ eRICTimer_UpMode+ eRICTimer_Reset+ eRICTimer_InterruptEnable);

eRIC_TimerA0_Count_Read(); This will return the current 16 bit timer count value.
eRIC_TimerA0_Count_Set(intcountnumber); This can set the timer counter value to any 16 bit number.

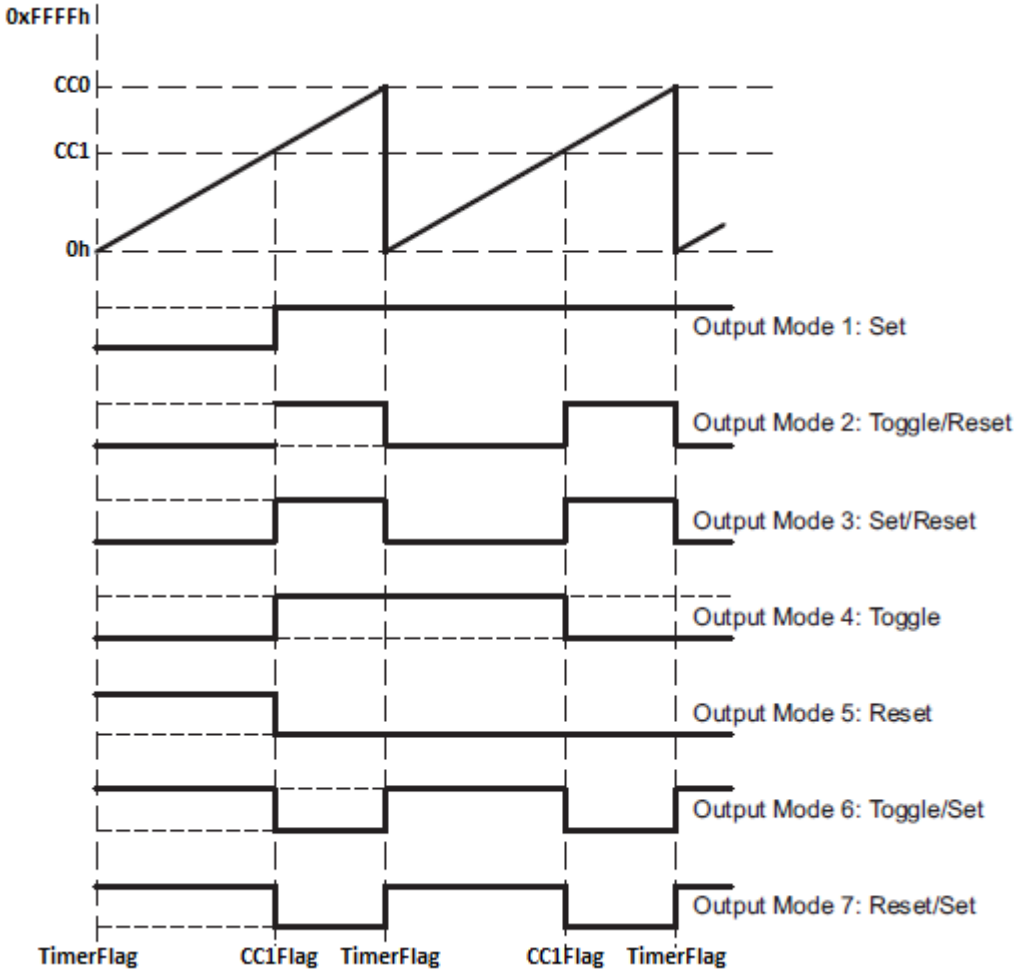
Setting TimerA0 Capture/Compare:

eRIC_TimerA0_CapturexorComparexSetup(CompleteSetup); This is used to setup five of the capture/compare individually. 'x' can be 0-4.
CompleteSetup can be sum of combination of below features:

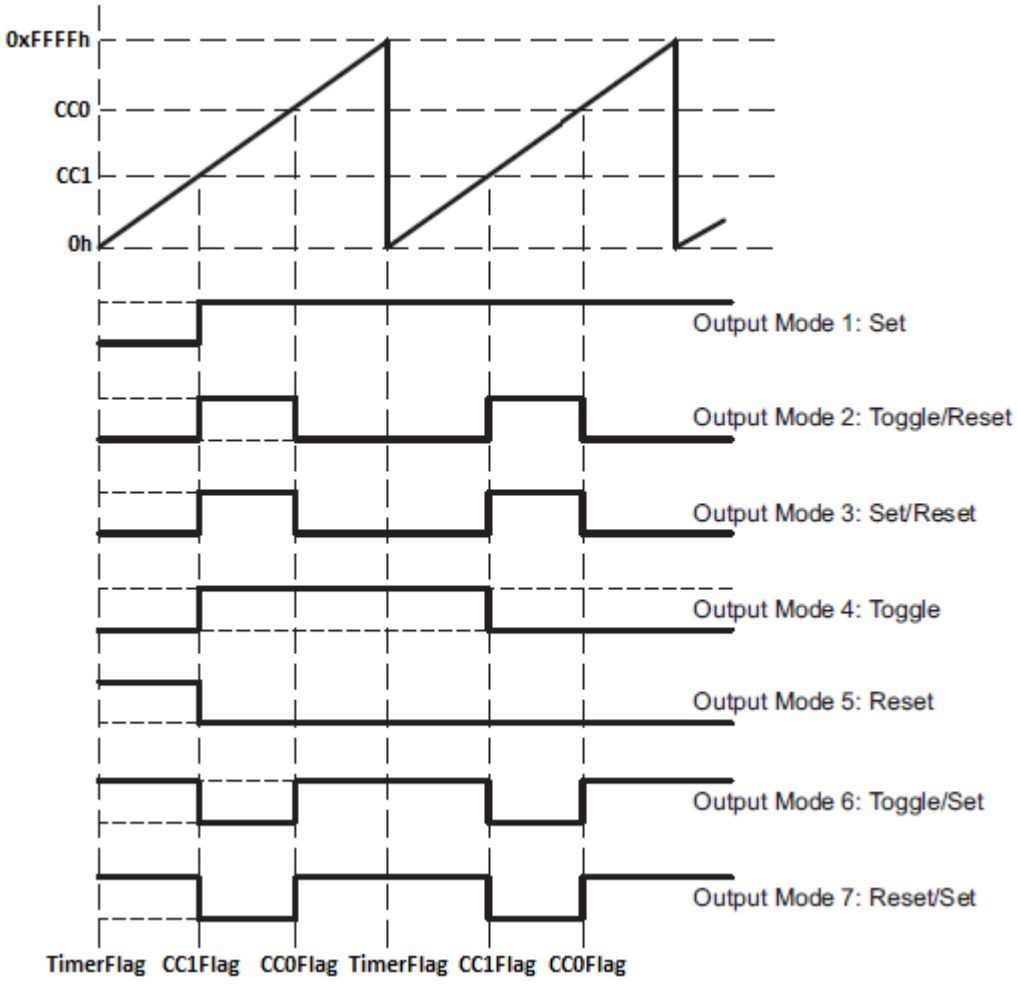
1) Capture or Compare Mode;
   a) eRICTimer_CaptureMode: This sets in Capture Mode
   e) eRICTimer_ComapreMode: This sets in Compare Mode.
2) Capture Modes:
   a) eRICTimer_CaptureNothing: No capture
   b) eRICTimer_CaptureOnRising: Capture on rising edge
   c) eRICTimer_CaptureOnFalling: Capture on falling edge
   d) eRICTimer_CaptureOnBothFallRise: Capture on both rising and falling edge.
3) Output Modes: Examples for this found below:
   a) eRICTimer_OutputMode_OutputOnly: Out bit value
   b) eRICTimer_OutputMode_Set: This set the capture/compare
   c) eRICTimer_OutputMode_Toggle_Reset: This toggles CCx and resets at CC0
   d) eRICTimer_OutputMode_Set_Reset: This sets at CCx and resets at CC0
   e) eRICTimer_OutputMode_Toggle: This toggles at CCx
   f) eRICTimer_OutputMode_Reset: This reset at CCx
   g) eRICTimer_OutputMode_Toggle_Set: This toggles at CCx and sets at CC0
   h) eRICTimer_OutputMode_Reset_Set: This resets at CCx and sets at CC0.
4) Interrupt:
   a) eRICTimer_CCInterruptEnable: This enables enables CCx interrupt. This is interrupt is different from main Timer interrupt.
   b) eRICTimer_CCInterruptDisable: This enables disables CCx interrupt. This is interrupt is different from main Timer interrupt.

OutPut Examples:

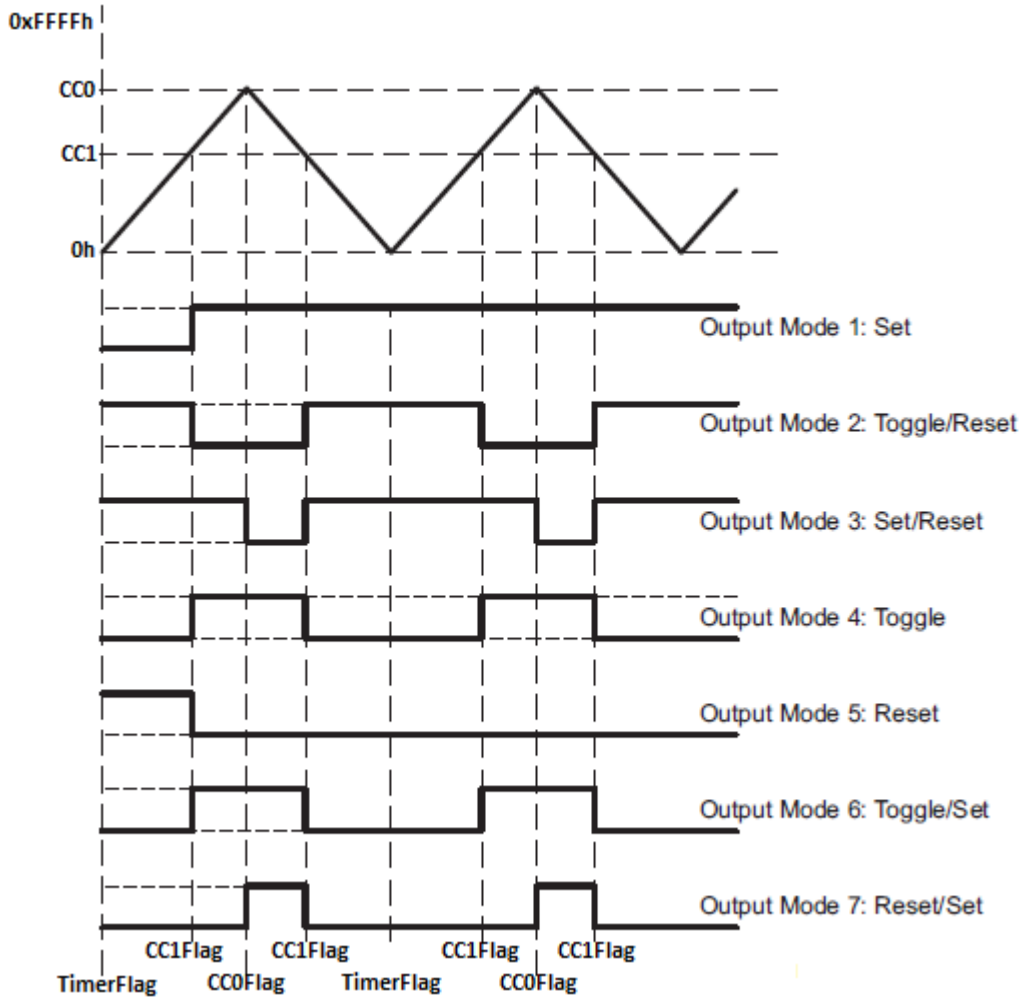
a) UpMode:



b) Continuous Mode:



c) UpdownMode:



eRIC_TimerA0_CapturexorComparex_Data; This is a 16 bit data. In compare mode, This(CCx) holds data for comparison to the timer count value. In capture mode, the timer count value is copied into this(CCx) when a capture is performed. 'x' is 0-4.

eRIC_TimerA0_CapturexorComparexInterruptEnable(); This enables CCx interrupt. This is different from main timer interrupt. 'x' is 0-4.

eRIC_TimerA0_CapturexorComparexInterruptDisable(); This disables CCx interrupt. This is different from main timer interrupt. 'x' is 0-4.

eRIC_TimerA0_CapturexorComparexInterruptFlagSet(); This sets CCx interrupt flag. This is different from main timer interrupt. 'x' is 0-4.

eRIC_TimerA0_CapturexorComparexInterruptFlagClear(); This clears CCx interrupt flag. This is different from main timer interrupt. 'x' is 0-4.

eRIC_TimerA0_CapturexorComparexHasInterrupted(); This can be used to check if CCx interrupt flag is set or cleared. This is different from main timer interrupt. 'x' is 0-4.

Code Example:

```
1) #include<cc430f5137.h>
2) #include "eRIC.h"
3) int main(void)
4) {
5)       eRIC_WDT_Stop();                  //Stop watch dog timer,just in case
6)       eRIC_GlobalInterruptDisable();    //Global interrupts disabled
7)       eRIC_SetCpuFrequency(1000000);    //Sets CPU clock to 1mHz
8)
9)       //TimerA0 is set with clock source of ACLK 32768 and Upmode(counts
   from 0-CC0)
10)      eRIC_TimerA0_Setup(eRICTimer_Cs_32k+eRICTimer_Div_1+eRICTimer_UpMode
   +eRICTimer_Reset);
11)      //Compare1 is set up with Outputmode setreset where CC1 pin will set
   at CC1 value and resets at CC0 value.
12)      eRIC_TimerA0_Capture1orCompare1Setup(eRICTimer_CompareMode+eRICTimer
   _CCInterruptEnable+eRICTimer_OutputMode_Set_Reset);
13)      eRIC_TimerA0_Capture0orCompare0Setup(eRICTimer_CompareMode+eRICTimer
   _CCInterruptEnable+eRICTimer_OutputMode_Set);
14)      eRIC_TimerA0_Capture0orCompare0_Data = 32768; //CC0 value
15)      eRIC_TimerA0_Capture1orCompare1_Data = 16384; //CC1 value
16)      Pin19_FunctionTA0CompareOut1();   //Assign to CC1 compare
17)      Pin18_SetAsOutput();
18)       Pin17_SetAsOutput();
19)      eRIC_GlobalInterruptEnable();
20)      while(1);
21)}//end of main()
22)
23)#pragma vector=TIMER0_A1_VECTOR
24)__interrupt void TIMER0__A1_ISR(void)
25){
26)  switch (TA0IV)
27)  {
28)  case TA0IV_TA0CCR1:
29)       Pin18_Toggle();
30)       break;
31)  default:
32)       break;
33)  }
34)}
35)#pragma vector=TIMER0_A0_VECTOR
36)__interrupt void TIMER0__A0_ISR(void)
37){
38)  Pin17_Toggle();
39)}
40)
41) void eRIC_RfDataReceivedInterrupt()   //V1.5.4 Add code here to deal with
   available received data..This is triggered when interrupt is enabled and a
   packet is received
42) {
43)  }
```

Line1 and line2 includes cc430F5137 and eRIC.h which is must for any program code. Main starts at line3.

Watchdogtimer is stoped in line4. Global interrupts are disabled in line5, any interrupts even radio interrupts in eros will be disabled.

TimerA0 is set in line10. 32768 clock which runs at 32768Hz is chosen. Clock Divider is set to 1 so clock is same. Mode is Upmode where timer counts from zero to the value of CC0 and start again from zero. It doesn't count to 0xFFFFh.

So when CC0 is 32768(which is what, it is chosen in code, in next step), the timer would take 1 sec to reach from 0 to 32768 as it uses ACLK(32768 cycles per second) as source.

Line12 CC1 is set as compare mode with output set at CC1 and reset at CC0. This is observed on Pin19 which is assigned to CC1 at line 16. In this same line 12 ,CC1 interrupt is also enabled which sets the flag and interrupts in vector A1 in line23.At line 28, is the flag for CC1 which is set when CC1 value is reached to 16384(this is set in line15). When CC1 flag is set, Pin18 is toggled in line29. It toggles every second, as CC1 flag is set every second according to the code.

At line13 CC0 is set as compare with output mode always set when it reached CC0 value. CC0 interrupt is also enabled which triggers the CC0 flag when CC0 value 32768(this is set at line14) is reached. When CC0 flag is set, the interrupt vector A0 is called and Pin17 is toggled accordingly. Even Pin17 should toggle every second as CC0 flag is set every second.

eRIC_RfDataReceivedInterrupt() is copied from eRIC.c which is removed and pasted in main at line 41-43. This has no effect in this example as there is no receiver enabled. But this code is needed for compiler to compile without error or un-remove this same code in eRIC.c.

The picture view of above logic is shown below: (not to scale)