Version 1.1

eRIC_PulseWidthModulation:

eRIC has four pulse width modulation which can be assigned to any of 18 pins. PWM is nothing but using TimerA0, with a period which is CC0 and four capture compare (CC1-CC4) as PWM1-PWM4. Refer TImerA0 before using this document.

Refer eRIC_eROS_Developers_Manual_x.x on wards for complete list of PWM definitions. Refer SLAU259E document by Texas Instruments to use core registers.

Setting PWM:

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

eRIC_PWM_Cs(Clocksource); PWM clock source can be set using this.
Clocksource can be:

a) eRICPWM_Cs_32k: This can be used to choose Aclk(32768) as clock source for Timer.
b) eRICPWM_Cs_CPU: This can be used to choose SMCLK(upto 20Mhz frequency set by eRIC_SetCpuFrequency()) as clock source for PWM.

eRIC_PWM_ClockDivider(Clockdivider); PWM clock can be further divided using this.
Clockdivider can be:

a) eRICPWM_Div_1: The clock source divided by 1
b) eRICPWM_Div_2: The clock source further divided by 2
c) eRICPWM_Div_4: The clock source further divided by 4
d) eRICPWM_Div_8: The clock source further divided by 8

eRIC_PWM_UpDownContinousMode(UpDownContinousMode); PWM have four different mode which can be chosen using UpDownContinousMode: UpDownContinousMode can be:

a) eRICPWM_Stop: This completely halts the Timer
b) eRICPWM_UpMode: Refer TimerA0 application note
c) eRICPWM_ContinousMode: Refer TimerA0 application note
d) eRICPWM_UpdownMode: Refer TimerA0 application note

eRIC_PWM_Reset(); Resets the count value of Timer

eRIC_PWM_Period(Period); Sets the period of PWM. Period can be any 16Bit value. This period is nothing but CC0. Refer TimerA0 application note.

All above settings can be done in one setup using this below:

eRIC_PWM_Setup(ClockSource,ClockDivider,UpDownContinousMode,Period); PWM can be set with all above individual bits in one function.

eRIC_PWMx_DutyCycle(DutyCycle,LogicOutput); Duty cycle of particular PWMx can be set by passing a 16 bit duty cycle value. 'x' can be 1,2,3,4 as only four PWM are available. LogicOutput can be any of the below:

a) eRICPWM_OutputMode_Set: Refer TimerA0 application note
b) eRICPWM_OutputMode_Toggle_Reset: Refer TimerA0 application note
c) eRICPWM_OutputMode_Set_Reset: Refer TimerA0 application note
d) eRICPWM_OutputMode_Toggle: Refer TimerA0 application note
e) eRICPWM_OutputMode_Reset: Refer TimerA0 application note
f) eRICPWM_OutputMode_Toggle_Set: Refer TimerA0 application note
g) eRICPWM_OutputMode_Reset_Set: Refer TimerA0 application note

Using Dutycycle, LogicOutput, clocksource, UPDownContinousModes and period, a PWM can be modified.

Code Example:

```
#include<cc430f5137.h>
#include "eRIC.h"
int main(void)
{
  eRIC_WDT_Stop();            //Stop watch dog timer, just in  case
  eRIC_GlobalInterruptDisable();     //Global interrupts disabled
  eRIC_SetCpuFrequency(1000000);    //Set Cpu frequency as 1mhz
  Pin19_FunctionPWM1();             //Assign Pin19 to PWM1
  Pin17_SetAsInput();               //Set Pin17 as input
  Pin17_PullDownEnable();
  Pin16_SetAsInput();               //Set Pin16 as input
  Pin16_PullDownEnable();
  eRIC_PWM_Setup(eRICPWM_Cs_32k,eRICPWM_DIV_1,eRICPWM_UpMode,512);    //Set
PWM with 32k clock source and Upmode and period being 512.
  int i=0;
  while(1)
  {
   while(Pin17_Read())
   {
    if(i<=512)
    eRIC_PWM1_DutyCycle(i++,eRICPWM_OutputMode_Reset_Set);
    eRIC_Delay(10);
   }
   while(Pin16_Read())
   {
    if(i>=0)
    eRIC_PWM1_DutyCycle(i--,eRICPWM_OutputMode_Reset_Set);
    eRIC_Delay(10);
   }
  }//end of while(1)
 }//end of main

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
 {
 }
```

The above code, will increase the duty cycle of PWM1 (Pin19) when Pin17 is high and decreases if Pin16 is high.

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

Watchdogtimer is stopped in line5. Global interrupts are disabled in line6, any interrupts even radio interrupts in eros will be disabled. Cpu clock is set to 1000000Hz in line7.

Pin19 is assigned as PWM1 in line8. Pin17 and 16 are set as inputs in line9 and 11 and are pulled low by using pull downs.

PWM is set with 32k clock source,clock divider as 1 and Upmode with period 512 in line13. So it takes

32768 cycles for 1 sec and for one period(512 cycles) it takes 15.625 milli seconds. There can be 64 periods in one second.

Variable ‘i’ is initialised as int in line14. The code loops continuously from line15 – line29.And if anytime Pin17 is held high, ‘i’ is increased to 512 which is the duty cycle ofPWM1 in steps of 1 for every 10 milliseconds, which is from line Line17-22. And if Pin16 is held high, ‘i’ is decreased to 0 which is duty cycle of PWM1 in steps of 1 for every 10 milliseconds which is from line 23-28. Changing duty cycle changes the pulse width.

eRIC_RfDataReceivedInterrupt() is copied from eRIC.c which is removed and pasted in main at line 38-40. 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.