```c
/* Beeper.c - implementation of Beeper.h
*
*     Copyright 2020 Graeme Judge, Sean Berkvens
*   Change Log:
*       May 3, 2020: Source file created
*/

#include "utils.h"
#include "Beeper.h"
#define PIN 6

void InitBeeper( void ){
  SET_BITS(RCC->AHB2ENR, RCC_AHB2ENR_GPIOBEN); //clock

  FORCE_BITS(GPIOB->MODER, 3UL << (2*PIN), 2UL << (2*PIN));
  FORCE_BITS(GPIOB->AFR[0], 0xF << (4 * PIN), 2UL << (4 * PIN));

  FORCE_BITS(GPIOB->PUPDR, 3UL << (2*PIN), 0);

  SET_BITS(RCC->APB1ENR1, RCC_APB1ENR1_TIM4EN);//time 4 clock

  CLR_BITS(TIM4-> CR1, TIM_CR1_DIR);  //up counting

  //TIM4->PSC = prescaleValue; --> 1MHz clock is 1us
  TIM4->PSC = 80-1;

  TIM4->ARR = 5000000 - 1; //auto reload every 0.5ms

  CLR_BITS(TIM4->CCMR1, TIM_CCMR1_OC1M);

  TIM4 -> CCR1 = 0; //Start without beeping

  SET_BITS(TIM4->BDTR, TIM_BDTR_MOE);   //output enable

  SET_BITS(TIM4->CCMR1, TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2);   //pwm

  CLR_BITS(TIM4 -> CCER, TIM_CCER_CC1P);        //active high
  SET_BITS(TIM4 -> CCER, TIM_CCER_CC1E);
  //start the counter
  SET_BITS(TIM4->CR1, TIM_CR1_CEN);
}

void Beep( uint32_t hertz ){
  SET_BITS(TIM4->BDTR, TIM_BDTR_MOE); //Turn beeper on
  SET_BITS(TIM4->CR1, TIM_CR1_CEN);

  //math things for the duty cycle
  uint32_t periodInUs = (1.0 / (double)hertz) * 1000000; //get uS period
  uint32_t autoReloadValue = periodInUs * 10 - 1;
  TIM4->ARR = autoReloadValue;
  TIM4 -> CCR1 = (TIM4->ARR + 1) / 2;
}

void StopBeep(){
  CLR_BITS(TIM4->CR1, TIM_CR1_CEN); //turns beeper off
  CLR_BITS(TIM4->BDTR, TIM_BDTR_MOE);
}
```