```c
// Link this source code with his .h file.
#include "timer05.h"

void config_timer05()
{
  /*
    config_UTS :: void -> void

    Configure the Timer0 to interrupt
    every 0.5 seconds.
  */

  LPC_SC->PCONP |=(1<<1);                             // Configure the power supply.
  LPC_TIM0->PR   = 0;                                 // No prescale -> 25MHz.
  LPC_TIM0->MR0  = (Fpclk*0.5-1);                     // Match at 0.5s.
  LPC_TIM0->MCR  = 3;                                 // When the time counter
  reachs the match interrupt,stop the TC and reset TC.
  LPC_TIM0->TCR |= (1<<0);                            // Start count.
  NVIC_EnableIRQ(TIMER0_IRQn);                        // Enables the interruption of
  Timer0.
}

void TIMER0_IRQHandler()
{
  /*
    TIMER0_IRQHandler :: void -> void

    Handles the interruption that is
    generated when the timer count up
    to 0.5 seconds.

    If the sonar is in Setup mode
    the mode is changed to manual.

    If the sonar is in automatic
    mode the UTS takes a measure,
    also if the servo is not blocked
    by the flag f_block_move and
    it's time to move servo, we
    move the servo.

    Instead, if the sonar is in the
    manual mode and the UTS is not
    blocked by the flag f_block_measure
    the UTS takes a measure.
  */

  static char sentido = POSITIVO;                     // Static variable that
  indicates the direction of the move.
  static int  cycle = 0;                              // Static variable that
  indicates how much cycles have
                                                      // passed since the last time
                                                      the servo was turned.

  LPC_TIM0->IR=1<<0;                                  // Clear the flag of the match
  interrupt,

  switch(sonar.state)
  {
    case(ST_SETUP):                                   // If the sonar is in Setup mode
      sonar.state = ST_MANUAL;                        // Change the mode to manual
      mode.
      break;

    case(ST_AUTOMATIC):                               // If the sonar is in
    Automatic mode:
      cycle++;                                        // Increase the number of cycles
      if(!sonar.f_block_measure)                      // If the UTS is allowed to
      measure:
        UTS_trigger();                                // Make a measure with the UTS

      if(!sonar.f_block_move                          // If the servo is allowed to
      move
```

```
     &&                    // AND
    (cycle >= sonar.servo_period))      // the cycle coincides with
    the servo's period.
   {
    if(sentido == POSITIVO)             // If the direction is positive.
    {
     if((sonar.servo_pose               // If the next move exceeds
     the bounds.
       +
       sonar.servo_resolution) > 180)
      sentido = NEGATIVO;               // Change the direction of the
      movement.

     else
      set_servo(sonar.servo_pose        // Increse the angle of the
      servo.
       += sonar.servo_resolution);
    }

    else                                // If the direction is negative.
    {
     if((sonar.servo_pose               // If the next move exceeds
     the bounds.
       -
       sonar.servo_resolution) < 0)
     {
      sentido = POSITIVO;               // Change the direction of the
      movement.
      set_servo(sonar.servo_pose        // Increase the angle.
       += sonar.servo_resolution);
     }
     else
      set_servo(sonar.servo_pose        // Decrease the angle of the
      servo.
       -= sonar.servo_resolution);
    }
    cycle = 0;                          // Reset the cycle counter.
   }
   break;

  case(ST_MANUAL):
   if(!sonar.f_block_measure)           // If the UTS is allowed to
   measure:
    UTS_trigger();                      // Make a measure with the UTS.
   break;
 }
}
```