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

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

    Enables the buttons
    and their interruptions.
  */

  LPC_PINCON->PINSEL4|=(1<<20)|(1<<22)|(1<<24);       // Configuration of pin functionality.
  NVIC_EnableIRQ(EINT0_IRQn);                         // Enable the interruption of ISP.
  NVIC_EnableIRQ(EINT1_IRQn);                         // Enable the interruption of KEY1.
  NVIC_EnableIRQ(EINT2_IRQn);                         // Enable the interruption of KEY2.
}


void EINT0_IRQHandler()
{
  /*
  EINT0_IRQHandler :: void -> void

  Handles the interruption that is
  generated when the ISP button
  is pressed.

  If the sonar is in automatic
  mode the flag f_block_move toggles,
  When this flag is high the servo
  stops moving.

  Instead, if the sonar is in manual
  mode the flag f_block_measure is
  the flag that toggles. When this
  flag is rised the UTS stops
  measuring.

  If the sonar is in another mode
  this function does not do
  anything.
  */

  LPC_SC->EXTINT |= (1<<0);                           // Clear the interruption flag.
  if(!sonar.f_block_keys)                             // Is the firt time that interrupts this cycle?
  {
    switch(sonar.state)
    {
      case(ST_AUTOMATIC):                             // If we are in aumatic mode:
        sonar.f_block_move ^= 1;                      // Toggle the f_block_move flag.
        break;
      case(ST_MANUAL):                                // If we are in manual mode:
        sonar.f_block_measure ^= 1;                   // Toggle the f_block_measure flag.
        break;
    }
    sonar.f_block_keys = 1;                           // Raise the flag to indicate that we have alredy
  }                                                   // interrupted this cycle.
}

void EINT1_IRQHandler()
{
   /*
    EINT1_IRQHandler :: void -> void

    Handles the interruption that is
```

```
67        generated when the KEY1 button
68        is pressed.
69
70        If the sonar is in Setup mode
71        the mode is changed to automatic,
72        and the UART is configured with
73        a baudrate of 9600 bauds.
74
75        Instead, if the sonar is in manual
76        mode the servo moves 10 degrees
77        in positive direction as long as
78        it not exceeds the maximun angle,
79        in our case 180 degrees.
80
81        If the sonar is in another mode
82        this function does not do
83        anything.
84      */
85
86      LPC_SC->EXTINT |= (1<<1);                       // Clear the interruption flag.
87      switch(sonar.state)
88      {
89        case(ST_SETUP):                               // If we are in Setup mode:
90          sonar.state = ST_AUTOMATIC;                 // Change the mode to
            automatic mode.
91          uart0_init(UART_BAUDRATE);                  // Configure the UART protocol.
92          sonar.f_block_measure  = 0;                 // Initialize the flag for
            automatic mode.
93
94          break;
95
96        case(ST_MANUAL):                              // If we are in Manual mode:
97          if(
98            (!((sonar.servo_pose + 10) > 180))        // If the sonar does not
              exceed the maximun angle
99            &&                                        // in the next move AND is the
              first time that
100           !(sonar.f_block_keys))                    // interrupts this cycle?
101         {
102           set_servo(sonar.servo_pose += 10);        // Increase the servo pose by
              10 degrees.
103           sonar.f_block_keys = 1;                   // Raise the flag to indicate
              that we have alredy
104         }                                           // interrupted this cycle.
105         break;
106     }
107   }
108
109   void EINT2_IRQHandler()
110   {
111     /*
112       EINT2_IRQHandler :: void -> void
113
114       Handles the interruption that is
115       generated when the KEY1 button
116       is pressed.
117
118       If the sonar is in manual
119       mode the servo moves 10 degrees
120       in negative direction as long as
121       it not exceeds the minimun angle,
122       in our case 0 degrees.
123
124       If the sonar is in another mode
125       this function does not do
126       anything.
127     */
128
129     LPC_SC->EXTINT |= (1<<2);                       // Clear the interruption flag.
130     if(
131       (sonar.state == ST_MANUAL)                    // If we are in Manual mode
132       &&                                            // AND
133       (!((sonar.servo_pose - 10) < 0))              // the sonar does not exceed
```

```
        the minimun angle in the next move
134       &&                                                  // AND
135       !(sonar.f_block_keys)                               // is the first time that
          interrupts this cycle
136     )
137     {
138       set_servo(sonar.servo_pose -= 10);                  // Decrease the servo pose by
          10 degrees.
139       sonar.f_block_keys = 1;                             // Raise the flag to indicate
          that we have alredy
140     }                                                     // interrupted this cycle.
141   }
142
```