```c

#include "spi_rfid.h"
#include "lcd.h"
#include "adc.h"



void initialize_SPI(void)
{
  RCC->APB2ENR |= RCC_APB2ENR_AFIOEN | RCC_APB2ENR_IOPBEN | RCC_APB2ENR_IOPCEN ;
  RCC->APB1ENR |= RCC_APB1ENR_SPI2EN;

  GPIOB->CRH |= GPIO_CRH_CNF15_1 | GPIO_CRH_MODE15;
  GPIOB->CRH &= ~GPIO_CRH_CNF15_0;

  GPIOB->CRH |= GPIO_CRH_CNF13_1 | GPIO_CRH_MODE13;
  GPIOB->CRH &= ~GPIO_CRH_CNF13_0;

  GPIOB->CRH |= GPIO_CRH_CNF14_1 | GPIO_CRH_MODE14;
  GPIOB->CRH &= ~GPIO_CRH_CNF14_0;

  GPIOB->CRH &= ~GPIO_CRH_CNF12;
  GPIOB->CRH |= GPIO_CRH_MODE12;


  SPI2->CR1 = 0x00;
  SPI2->CR1 |= SPI_CR1_SSM | SPI_CR1_CPOL | SPI_CR1_LSBFIRST | SPI_CR1_SSI;
  SPI2->CR2 &= ~SPI_CR2_SSOE;
  SPI2->CR1 |= SPI_CR1_BR_2 | SPI_CR1_BR_1;
  SPI2->CR1 |= SPI_CR1_SPE | SPI_CR1_MSTR;

  slave_select(true);
  delay_ms(1);
  uint8_t data[64];
  data[0] = 0x02;
  read_target(data, 1, 1000);
  data[0] = 0x14;
  data[1] = 0x01;
  data[2] = 0x14;
  data[3] = 0x01;
  read_target(data, 4, 1000);
  delay_ms(1);
  slave_select(false);

}
void send_byte(uint8_t byte)
{

  while((SPI2->SR & SPI_SR_TXE) != 0x2){}
  SPI2->DR = byte;
  while((SPI2->SR & SPI_SR_RXNE) != 0x1){}
  while (SPI2->SR & (SPI_I2S_FLAG_BSY)){}
  int z = SPI2->DR;
}
uint8_t receive_byte(void)
{
  SPI2->DR = 0x00;
  while((SPI2->SR & SPI_SR_TXE) != 0x2){}

  while((SPI2->SR & SPI_SR_RXNE) != 0x1){}
  while (SPI2->SR & (SPI_I2S_FLAG_BSY));
  return SPI2->DR;
}
void slave_select(bool set)
{
  if (set)
  {
    GPIOB->BRR |= GPIO_BRR_BR12;
  }
  else
  {
    GPIOB->BSRR |= GPIO_BSRR_BS12;
  }
```

```
  }
  bool IRQ_ready(void)
  {
    if ((GPIOB->IDR & GPIO_IDR_IDR11) != 0)
    {
      return false;
    }
    return true;
  }
  bool RFID_write_ack(uint8_t *data, uint8_t cmd_len, uint32_t timeout)
  {
    RFID_write(data, cmd_len);
    if (!ready_timed(timeout))
    {
      return false;
    }
    uint8_t temp_buff[6];
    const uint8_t buff[] = {0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00};
    RFID_read(temp_buff, 6);
    int val = strcmp((char*)buff, (char*)temp_buff);
    if (val != 0)
    {
      return false;
    }
    if(!ready_timed(timeout))
    {
      return false;
    }

    return true;
  }
  void RFID_write(uint8_t *data, uint8_t cmd_len)
  {
    uint8_t check_sum;
    cmd_len++;
    slave_select(true);
    delay_ms(1);
    send_byte(0x01); //data writing
    send_byte(0x00); //preamble
    send_byte(0x00); //preamble
    send_byte(0xFF); //start code 2
    send_byte(cmd_len); //TFI + # of data packets
    send_byte(~cmd_len + 1); //so cmd_len + ~cmd_len + 1 == 0
    send_byte(0xD4);
    check_sum = 0xFF + 0xD4;  //Start code + TFI
    for (int i = 0; i < cmd_len-1; i++)
    {
      send_byte(data[i]);
      check_sum += data[i]; //Start code + TFI + data bytes in lowest byte of check sum
    }
    send_byte(~check_sum);
    send_byte(0x00);
    slave_select(false);
    delay_ms(1);
  }
  bool is_ready(void)
  {
    if(!IRQ_ready())
    {
      return false;
    }
    return true;
  }
  void RFID_read(uint8_t *x, uint8_t len)
  {
    slave_select(true);
    delay_ms(1);
    send_byte(0x03); //data read
    for (int i = 0; i < len; i++)
    {
      delay_ms(1);
      x[i] = receive_byte();

```

```
147     }
148     slave_select(false);
149 
150   }
151 
152 
153 
154   bool ready_timed(uint32_t timeout)
155   {
156     set_time_out(timeout);
157     while(!timed_out())
158     {
159       if(is_ready())
160       {
161         return true;
162       }
163     }
164     return false;
165   }
166   bool read_target(uint8_t *data, uint8_t len, uint32_t timeout)
167   {
168 
169 
170        bool val = RFID_write_ack(data, len, timeout);
171        //uint8_t x[15];
172         if (!val)
173           return false;
174        RFID_read(data, 20);
175        CMD_2_LCD(LCD_LN1);
176        return true;
177 
178   }
179   uint32_t rfid_get_data(void)
180   {
181        initialize_SPI();
182        uint8_t data[64];
183        CMD_2_LCD(LCD_CLR);
184        data[0] = 0x4A;
185        data[1] = 0x01;
186        data[2] = 0x00;
187        second_2_LCD();
188        bool val = read_target(data, 3, 1000000);
189        if (!val)
190          return 0x00000000;
191        CMD_2_LCD(LCD_CLR);
192        CMD_2_LCD(LCD_LN2);
193         int ax = 0;
194         for(int i = 0; i < 12; i++)
195         {
196         if (data[i] != 0xD5)
197           ax++;
198         else if (data[i] == 0xD5)
199           break;
200         }
201         int a = 0;
202         uint32_t temp_2;
203        for (int z = ax+8; z < ax+12; z++)
204        {
205         uint32_t temp;
206         temp = data[z];
207 
208         temp_2 += (temp << (24 - a));
209         a += 8;
210        }
211       return temp_2;
212 
213   }
214 
```