ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KĨ THUẬT MÁY TÍNH

BÁO CÁO QUÁ TRÌNH
THỰC TẬP DOANH NGHIỆP

1. YÊU CẦU

1.1. Gửi dữ liệu qua USART

1.2. Nhận dữ liệu qua USART

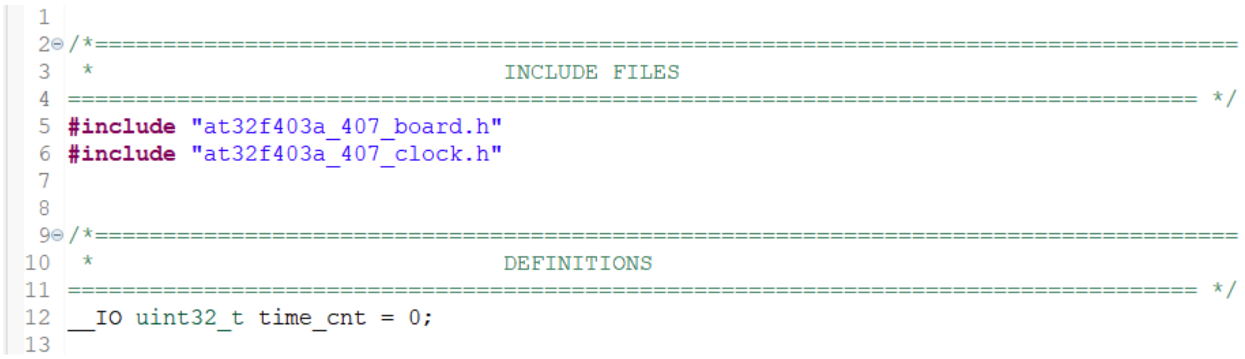
2. Gửi dữ liệu

2.1. README

- Send string through USART1, baudrate 115200
- Notice:
- AT32 IDE and Hercules terminal is recommended
- AT-START-F403A is currently in use

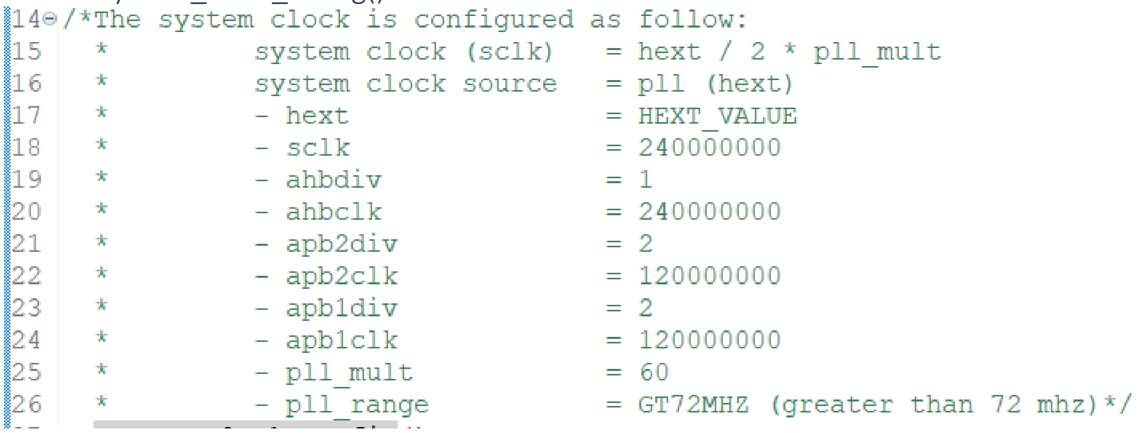
2.2. Source code

2.2.1. Include & definition



```c

/*==============================================================================
 *                              INCLUDE FILES
============================================================================= */
#include "at32f403a_407_board.h"
#include "at32f403a_407_clock.h"


/*==============================================================================
 *                              DEFINITIONS
============================================================================= */
__IO uint32_t time_cnt = 0;

```

2.2.2. system_clock_config():



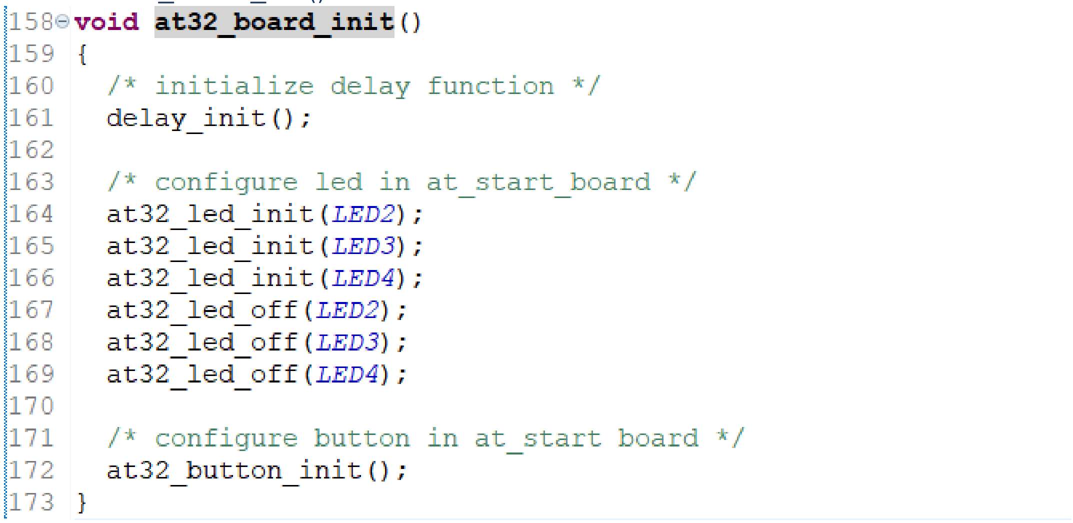
```c
/*The system clock is configured as follow:
 *          system clock (sclk)    = hext / 2 * pll_mult
 *          system clock source    = pll (hext)
 *          - hext                 = HEXT_VALUE
 *          - sclk                 = 240000000
 *          - ahbdiv               = 1
 *          - ahbclk               = 240000000
 *          - apb2div              = 2
 *          - apb2clk              = 120000000
 *          - apb1div              = 2
 *          - apb1clk              = 120000000
 *          - pll_mult             = 60
 *          - pll_range            = GT72MHZ (greater than 72 mhz)*/
```

- ○ system clock source (= 240 000 000) = pll (hext): clock ngoại thạch anh (HEXT_VALUE = 8 000 000) * pll_mult(=60)
- ○ => PLL_range greater than 72mhz
- ○ Apb1clk (= sclk / apb1div(=2)): 120 000 000 (maximum frequency of APB1/APB2)

```c
void system_core_clock_update(void)
```

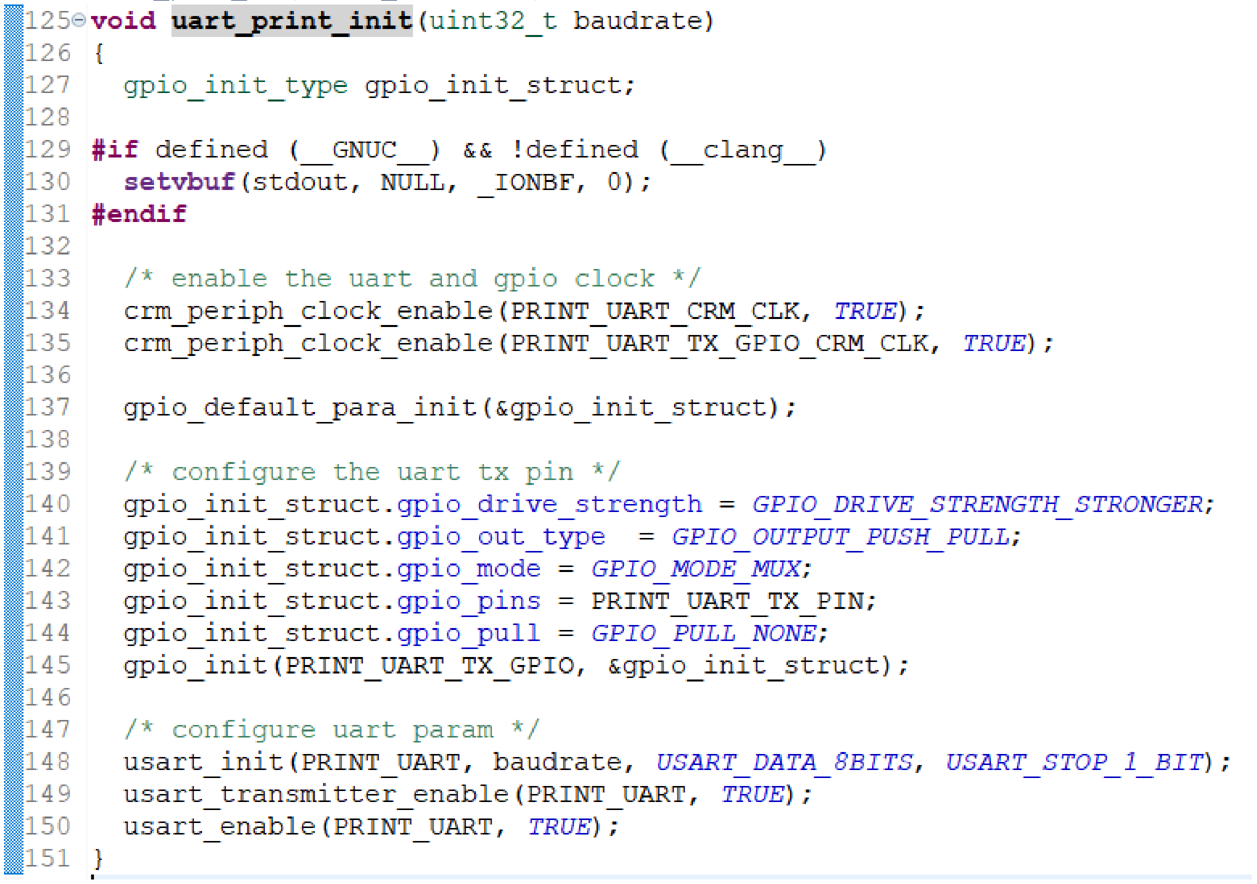
```
{
  uint32_t hext_prediv = 0, pll_mult = 0, pll_mult_h = 0, pll_clock_source =
0, temp = 0, div_value = 0;
  crm_sclk_type sclk_source;

  static const uint8_t sys_ahb_div_table[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2,
3, 4, 6, 7, 8, 9};

  /* get sclk source */
  sclk_source = crm_sysclk_switch_status_get();

  switch(sclk_source)
  {
    case CRM_SCLK_HICK:
      if(((CRM->misc3_bit.hick_to_sclk) != RESET) && ((CRM-
>misc1_bit.hickdiv) != RESET))
        system_core_clock = HICK_VALUE * 6;
      else
        system_core_clock = HICK_VALUE;
      break;
    case CRM_SCLK_HEXT:
      system_core_clock = HEXT_VALUE;
      break;
    case CRM_SCLK_PLL:
      pll_clock_source = CRM->cfg_bit.pllrcs;
      {
        /* get multiplication factor */
        pll_mult = CRM->cfg_bit.pllmult_l;
        pll_mult_h = CRM->cfg_bit.pllmult_h;
        /* process high bits */
        if((pll_mult_h != 0U) || (pll_mult == 15U)){
            pll_mult += ((16U * pll_mult_h) + 1U);
        }
        else
        {
            pll_mult += 2U;
        }

        if (pll_clock_source == 0x00)
        {
          /* hick divided by 2 selected as pll clock entry */
          system_core_clock = (HICK_VALUE >> 1) * pll_mult;
        }
        else
        {
          /* hext selected as pll clock entry */
          if (CRM->cfg_bit.pllhextdiv != RESET)
          {
            hext_prediv = CRM->misc3_bit.hextdiv;

            /* hext clock divided by 2 */
            system_core_clock = (HEXT_VALUE / (hext_prediv + 2)) * pll_mult;
          }
          else
          {
            system_core_clock = HEXT_VALUE * pll_mult;
          }
        }
      }
      break;
    default:
      system_core_clock = HICK_VALUE;
      break;
  }
```

2.2.3. at32_board_init():



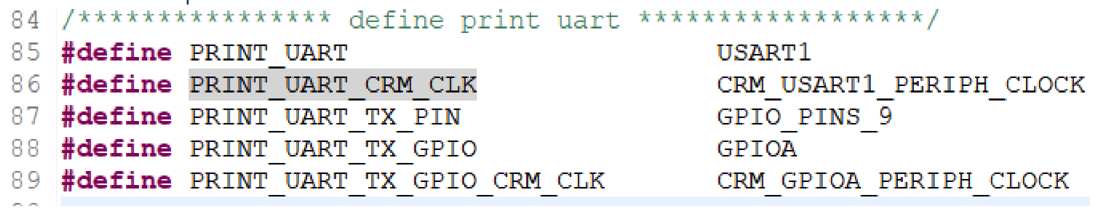
```
void at32_board_init()
{
  /* initialize delay function */
  delay_init();

  /* configure led in at_start_board */
  at32_led_init(LED2);
  at32_led_init(LED3);
  at32_led_init(LED4);
  at32_led_off(LED2);
  at32_led_off(LED3);
  at32_led_off(LED4);

  /* configure button in at_start board */
  at32_button_init();
}
```

2.2.4. uart_print_init(uint32_t baudrate)



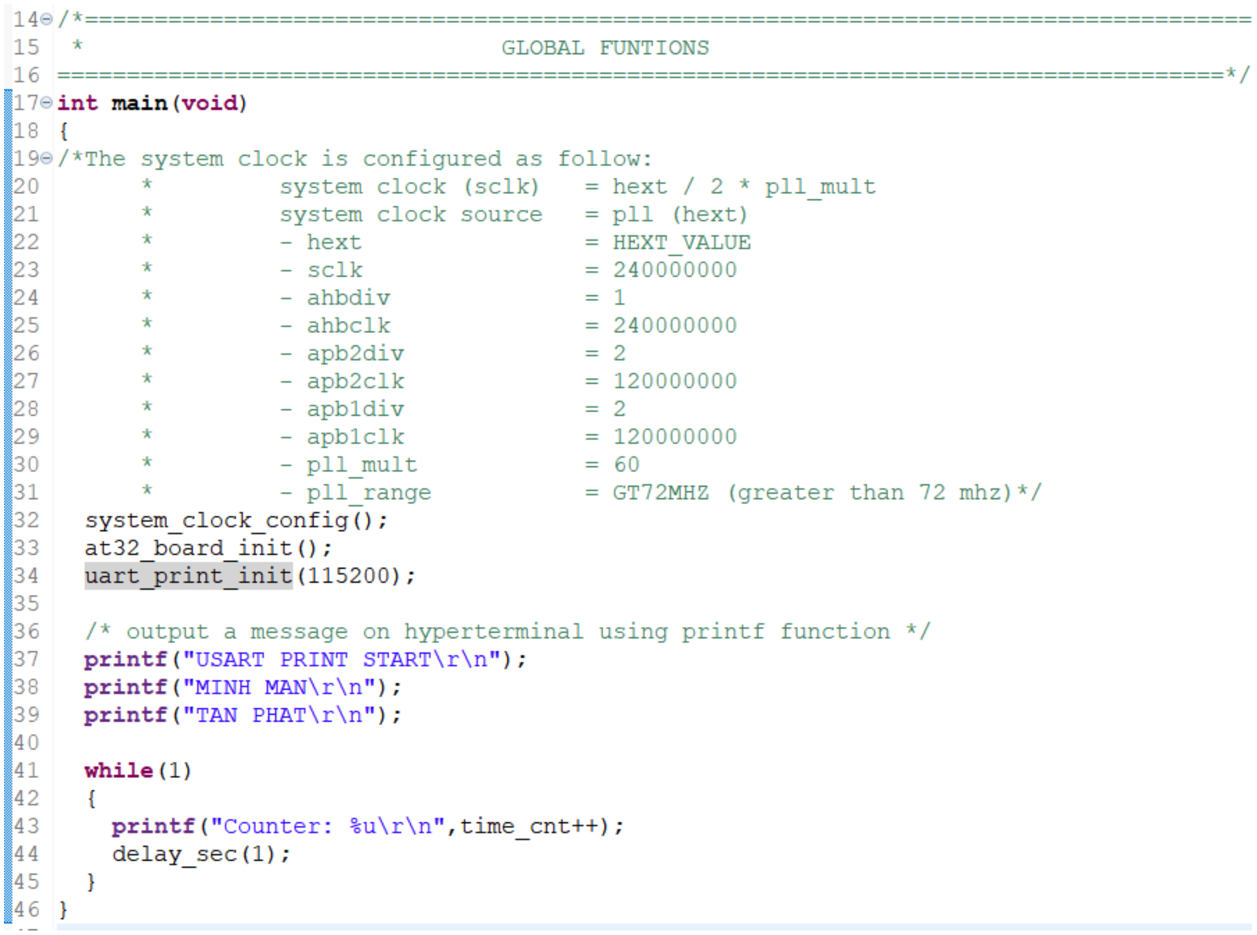
```
void uart_print_init(uint32_t baudrate)
{
  gpio_init_type gpio_init_struct;

#if defined (__GNUC__) && !defined (__clang__)
  setvbuf(stdout, NULL, _IONBF, 0);
#endif

  /* enable the uart and gpio clock */
  crm_periph_clock_enable(PRINT_UART_CRM_CLK, TRUE);
  crm_periph_clock_enable(PRINT_UART_TX_GPIO_CRM_CLK, TRUE);

  gpio_default_para_init(&gpio_init_struct);

  /* configure the uart tx pin */
  gpio_init_struct.gpio_drive_strength = GPIO_DRIVE_STRENGTH_STRONGER;
  gpio_init_struct.gpio_out_type  = GPIO_OUTPUT_PUSH_PULL;
  gpio_init_struct.gpio_mode = GPIO_MODE_MUX;
  gpio_init_struct.gpio_pins = PRINT_UART_TX_PIN;
  gpio_init_struct.gpio_pull = GPIO_PULL_NONE;
  gpio_init(PRINT_UART_TX_GPIO, &gpio_init_struct);

  /* configure uart param */
  usart_init(PRINT_UART, baudrate, USART_DATA_8BITS, USART_STOP_1_BIT);
  usart_transmitter_enable(PRINT_UART, TRUE);
  usart_enable(PRINT_UART, TRUE);
}
```

2.2.5. Define print uart



```c
/***************** define print uart ******************/
#define PRINT_UART                       USART1
#define PRINT_UART_CRM_CLK               CRM_USART1_PERIPH_CLOCK
#define PRINT_UART_TX_PIN                GPIO_PINS_9
#define PRINT_UART_TX_GPIO               GPIOA
#define PRINT_UART_TX_GPIO_CRM_CLK       CRM_GPIOA_PERIPH_CLOCK
```

2.2.6. Main code



```c
/*==========================================================================================
 *                              GLOBAL FUNTIONS
 ==========================================================================================*/
int main(void)
{
/*The system clock is configured as follow:
      *          system clock (sclk)    = hext / 2 * pll_mult
      *          system clock source    = pll (hext)
      *          - hext                 = HEXT_VALUE
      *          - sclk                 = 240000000
      *          - ahbdiv               = 1
      *          - ahbclk               = 240000000
      *          - apb2div              = 2
      *          - apb2clk              = 120000000
      *          - apb1div              = 2
      *          - apb1clk              = 120000000
      *          - pll_mult             = 60
      *          - pll_range            = GT72MHZ (greater than 72 mhz)*/
  system_clock_config();
  at32_board_init();
  uart_print_init(115200);

  /* output a message on hyperterminal using printf function */
  printf("USART PRINT START\r\n");
  printf("MINH MAN\r\n");
  printf("TAN PHAT\r\n");

  while(1)
  {
    printf("Counter: %u\r\n",time_cnt++);
    delay_sec(1);
  }
}
```

- Gửi USART PRINT START\r\n"MINH MAN\r\n"TAN PHAT\r\n" khi reset
- Gửi giá trị Counter tăng dần sau mỗi 1 giây

2.3. Github & Video demo

2.3.1. Github:

Github_print

2.3.2. Video demo:

USART_print

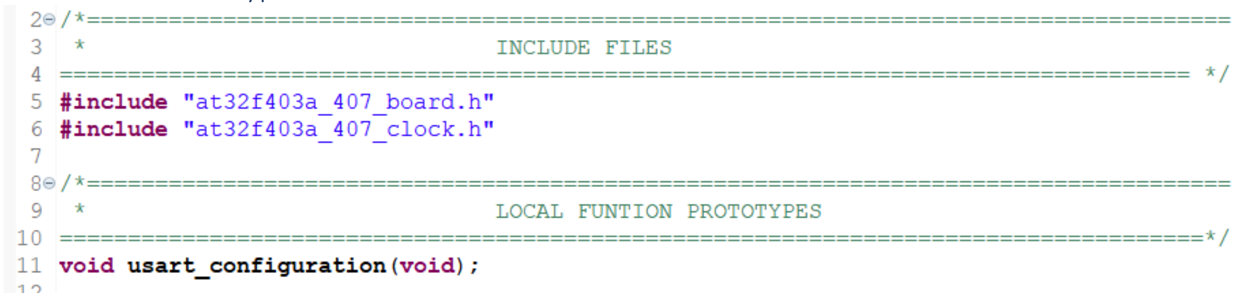
3. Nhận dữ liệu

3.1. README

- Control LED2,3,4 when get accurate data
- LED2: 0x02
- LED3: 0x03
- LED4: 0x04
- Notice:
- AT32 IDE and Hercules terminal is recommended
- AT-START-F403A is currently in use

3.2. Source code

3.2.1. Include & Typedef

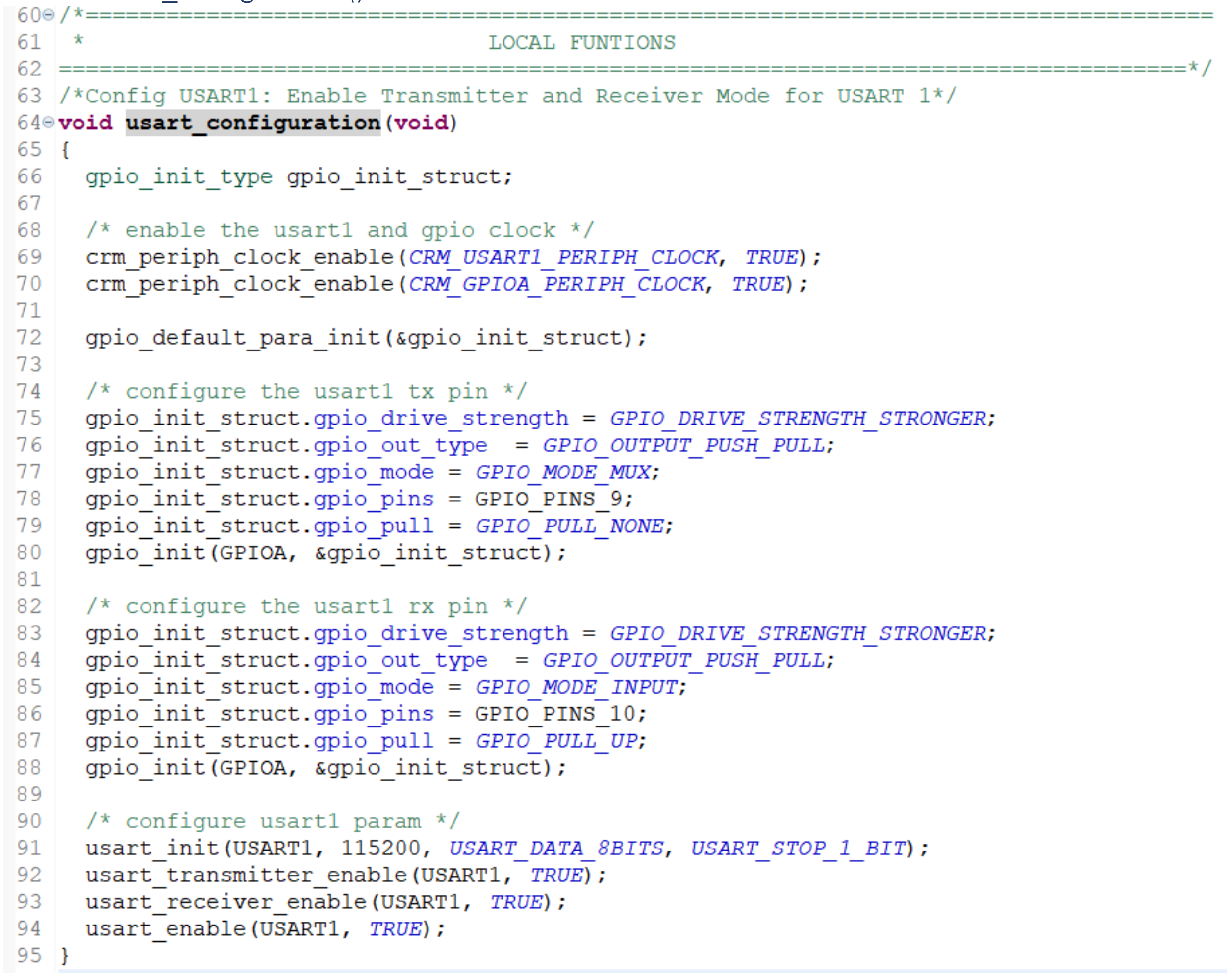


```
/*==========================================================================
 *                              INCLUDE FILES
========================================================================== */
#include "at32f403a_407_board.h"
#include "at32f403a_407_clock.h"

/*==========================================================================
 *                              LOCAL FUNTION PROTOTYPES
==========================================================================*/
void usart_configuration(void);
```

3.2.2. system_clock_config(); at32_board_init();

- Tương tự config Gửi dữ liệu

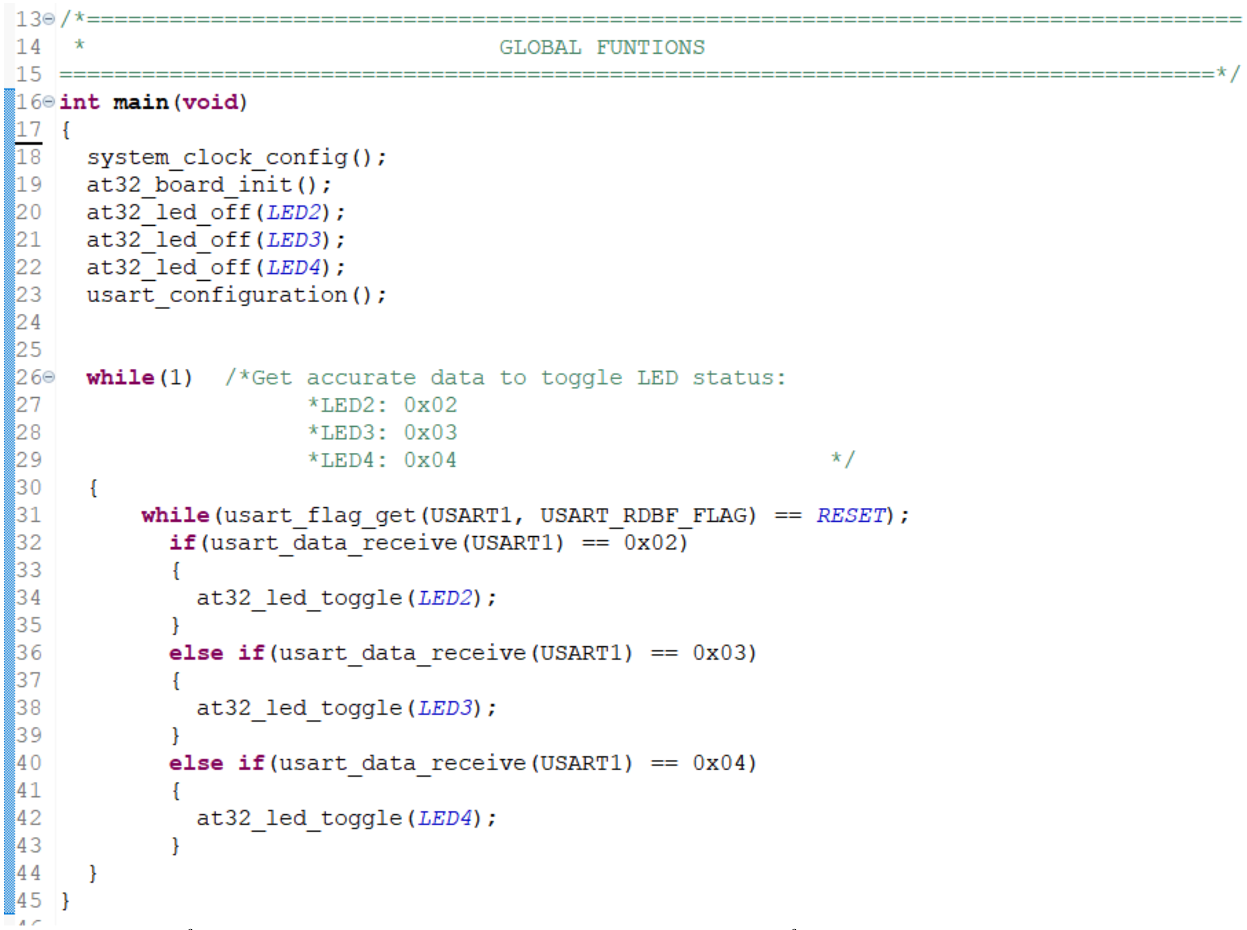
3.2.3. usart_configuration()



```c
/*=================================================================================
 *                               LOCAL FUNTIONS
=================================================================================*/
/*Config USART1: Enable Transmitter and Receiver Mode for USART 1*/
void usart_configuration(void)
{
  gpio_init_type gpio_init_struct;

  /* enable the usart1 and gpio clock */
  crm_periph_clock_enable(CRM_USART1_PERIPH_CLOCK, TRUE);
  crm_periph_clock_enable(CRM_GPIOA_PERIPH_CLOCK, TRUE);

  gpio_default_para_init(&gpio_init_struct);

  /* configure the usart1 tx pin */
  gpio_init_struct.gpio_drive_strength = GPIO_DRIVE_STRENGTH_STRONGER;
  gpio_init_struct.gpio_out_type  = GPIO_OUTPUT_PUSH_PULL;
  gpio_init_struct.gpio_mode = GPIO_MODE_MUX;
  gpio_init_struct.gpio_pins = GPIO_PINS_9;
  gpio_init_struct.gpio_pull = GPIO_PULL_NONE;
  gpio_init(GPIOA, &gpio_init_struct);

  /* configure the usart1 rx pin */
  gpio_init_struct.gpio_drive_strength = GPIO_DRIVE_STRENGTH_STRONGER;
  gpio_init_struct.gpio_out_type  = GPIO_OUTPUT_PUSH_PULL;
  gpio_init_struct.gpio_mode = GPIO_MODE_INPUT;
  gpio_init_struct.gpio_pins = GPIO_PINS_10;
  gpio_init_struct.gpio_pull = GPIO_PULL_UP;
  gpio_init(GPIOA, &gpio_init_struct);

  /* configure usart1 param */
  usart_init(USART1, 115200, USART_DATA_8BITS, USART_STOP_1_BIT);
  usart_transmitter_enable(USART1, TRUE);
  usart_receiver_enable(USART1, TRUE);
  usart_enable(USART1, TRUE);
}
```

- Cấu hình GPIO, mode cho USART1 tx, rx
- Enable transmitter và receiver để gửi/ nhận dữ liệu

3.2.4. Main code



```c
/*===================================================================================
 *                                    GLOBAL FUNTIONS
===================================================================================*/
int main(void)
{
  system_clock_config();
  at32_board_init();
  at32_led_off(LED2);
  at32_led_off(LED3);
  at32_led_off(LED4);
  usart_configuration();


  while(1)  /*Get accurate data to toggle LED status:
                  *LED2: 0x02
                  *LED3: 0x03
                  *LED4: 0x04                           */
  {
      while(usart_flag_get(USART1, USART_RDBF_FLAG) == RESET);
        if(usart_data_receive(USART1) == 0x02)
        {
          at32_led_toggle(LED2);
        }
        else if(usart_data_receive(USART1) == 0x03)
        {
          at32_led_toggle(LED3);
        }
        else if(usart_data_receive(USART1) == 0x04)
        {
          at32_led_toggle(LED4);
        }
  }
}
```

- Thay đổi trạng thái LED khi nhận chính xác dữ liệu. Cụ thể:
    - 0x02: LED2
    - 0x03: LED3
    - 0x04: LED4

3.3. Github & Video demo:

3.3.1. Github:

Github_receive

3.3.2. Video demo:

USART_receive