



راه اندازی ارتباط پورت سریال UART به روش وقفه در STM32



تاریخ انتشار: ۳۰ بهمن، ۱۴۰۲ توسط سید حسین سلطانی

سلام خدمت همه شما مایکروالکامی ها. در مطلب قبلی از سری مطالب **STM32** به نحوه **دریافت دیتا از روی پورت سریال UART** پرداخته شد. در این مطلب به بررسی ارسال و دریافت روش وقفه (Interrupt) پورت سریال در حالت **non-blocking mode** با استفاده از توابع HAL در میکروکنترلر STM32 پرداخته خواهد شد. پس با من تا انتهای مطلب همراه باشید. همچنین شما میتونید سایر مطالب من رو از **این لینک** مطالعه و بررسی کنید.



مقدمه

دو روش کلی blocking و non-blocking mode در راه اندازی و استفاده از پورت سریال (UART) در STM32 وجود دارد. در روش اول CPU میکروکنترلر در زمان ارسال یا دریافت درگیر بوده و معایب خود را دارد. در non-blocking که شامل روش های وقفه (Interrupt) و DMA می شود این عیب دیگر وجود ندارد. در این مطلب به بررسی روش ارسال و دریافت از طریق وقفه سریال UART پرداخته خواهد شد.

ارتباط سریال UART در میکروکنترلر STM32

در مطالب قبلی به بررسی و راه اندازی ارتباط پورت سریال UART در STM32 پرداخته شد. طی دو مطلب به صورت مجزا نحوه ارسال دیتا و دریافت بررسی شد. برای بررسی و مطالعه بیشتر می توانید از طریق لینک های زیر استفاده کنید.

- [ارسال دیتا در ارتباط سریال به روش blocking](#)
- [دریافت دیتا از طریق پورت سریال به روش blocking](#)

بررسی روش وقفه در ارتباط پورت سریال STM32

در روش blocking از آنجا که timeout جهت ارسال یا دریافت دیتا تعریف می شود، باعث درگیری CPU برای انجام این کار شده که در اکثر موارد مشکل ساز خواهد شد. لذا برای رفع آن می توان از حالت non-blocking یا روش DMA و وقفه (Interrupt) برای ارسال و دریافت استفاده کرد.

روش وقفه

در روش وقفه (Interrupt) مراحل انجام کار بدون درگیری یا در پس زمینه کار انجام می پذیرند. لذا از همین رو مابقی دستورات و پردازش ها همانگونه که باید انجام خواهند شد و هنگام فراخوانی تابع، به سرویس روتین وقفه (ISR) وارد



شده و می‌توان از آن استفاده کرد. برای استفاده از روش وقفه، هنگام پیکربندی اولیه باید آنرا فراخوانی کرد که در ادامه بیان خواهد شد.

می‌توان از دو تابع سرویس روتین وقفه دریافت یا ارسال برای مطلع شدن از دریافت یا ارسال دیتا استفاده کرد. برای دریافت دیتا هنگام دریافت یک بایت در سریال، پس از وارد شدن به تابع ISR نسبت به ذخیره دیتا در یک آرایه اقدام می‌کنیم.

دریافت دیتا از سریال UART

برای دریافت به روش وقفه از دستور زیر می‌توان استفاده کرد.

```
HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t  
Size);
```

- ورودی اول: اشاره گر (pointer) به ساختار تنظیمات UART میکروکنترلر
- ورودی دوم: اشاره گر (pointer) از نوع uint8_t به آرایه محل ذخیره سازی دیتا مورد نظر
- ورودی سوم: یک عدد uint16_t بیانگر طول دیتای مورد نظر جهت دریافت که می‌توان از توابع sizeof() یا strlen() استفاده کرد.

ارسال دیتا با سریال UART

برای ارسال دیتا به روش وقفه از دستور زیر می‌توان استفاده کرد.

```
HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t  
Size);
```

- ورودی اول: اشاره گر (pointer) به ساختار تنظیمات UART میکروکنترلر
- ورودی دوم: اشاره گر (pointer) از نوع uint8_t به آرایه دیتا مورد نظر برای ارسال
- ورودی سوم: یک عدد uint16_t بیانگر طول دیتای مورد نظر جهت دریافت که می‌توان از توابع sizeof() یا strlen() استفاده کرد.



در صورت ارسال دیتا به روش وقفه، پس از اتمام فرآیند ارسال تابع زیر فراخوانی می‌شود. این تابع را باید در فایل main.c اضافه کرد.

```
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
    // write some code when transmission is complete
}
```

در صورت اتمام فرآیند دریافت دیتا، وقفه دریافت فراخوانی شده و وارد تابع آن می‌شود. این تابع را باید در فایل main.c اضافه کرد.

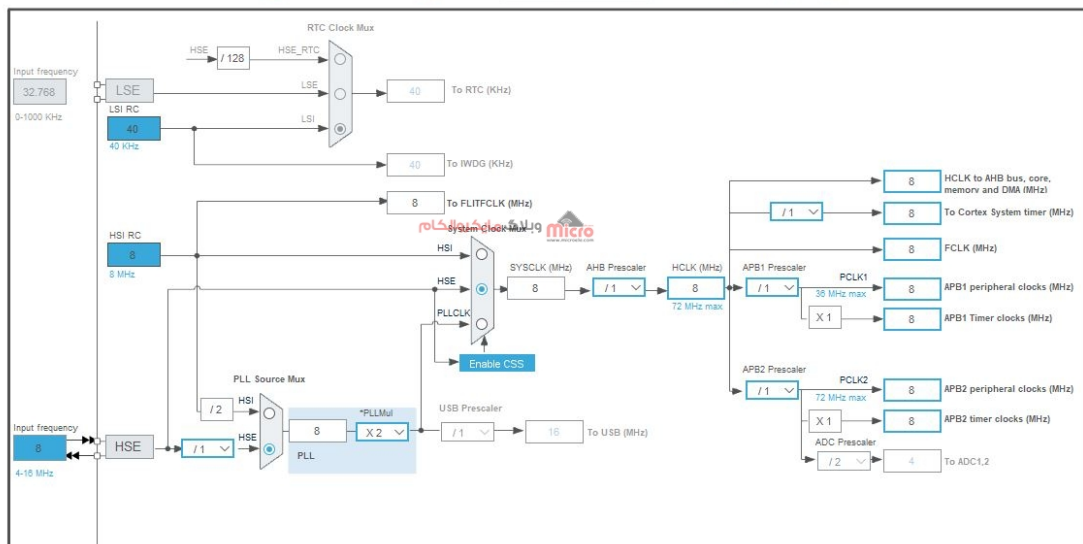
```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // write some code when reception is complete
}
```

پیکربندی و ایجاد پروژه

در ابتدا یک پروژه جدید ایجاد کرده و میکروکنترلر مورد نظر (در این مطلب STM32F103C8T6) را انتخاب و تنظیمات آن را مطابق مراحل زیر انجام می‌دهیم.

تنظیمات کلاک

بعد از باز شدن پنجره Mx از بخش System Core وارد RCC شده و منبع کلاک (HSE) را کریستال خارجی انتخاب و نهایتاً در بخش Clock Configuration فرکانس کاری میکرو را مطابق با نیاز انتخاب نمایید.



تنظیم کلاک در STM32 برای ارتباط سریال

تنظیمات GPIO

در Pinout & Configuration وارد System Core و GPIO شده و یک پایه را به دلخواه برای اتصال یک LED و چشمک زدن آن انتخاب می‌کنیم. در تصویر زیر PC13 انتخاب شده و یک label با نام LED به آن اختصاص داده شده است.



The screenshot shows the STM32CubeIDE interface for configuring the USART peripheral. The 'Pinout & Configuration' window is open, showing the 'Configuration' tab for the USART peripheral. The 'Pinout view' is active, displaying the pin connections for the USART1_TX and USART1_RX pins. The pin connections are as follows:

Pin	Signal	GPIO o	GPIO m	GPIO P	Maximu	User La	Modified
PC13-T...	n/a	Low	Output ...	No pull-...	Low	LED	<input checked="" type="checkbox"/>

The pinout view shows the following connections:

- PC13-T... (LED) connected to USART1_TX
- PC14... connected to USART1_RX
- PC15... connected to USART1_TX
- PD0... connected to USART1_RX
- PD1... connected to USART1_TX
- NRST connected to USART1_TX
- VSSA connected to USART1_RX
- VDDA connected to USART1_TX
- PA0... connected to USART1_RX
- PA1 connected to USART1_TX
- PA2 connected to USART1_RX
- PA3 connected to USART1_TX
- PA4 connected to USART1_RX
- PA5 connected to USART1_TX
- PA6 connected to USART1_RX
- PA7 connected to USART1_TX
- PA8 connected to USART1_RX
- PA9 connected to USART1_TX
- PA10 connected to USART1_RX
- PA11 connected to USART1_TX
- PA12 connected to USART1_RX
- PA13 connected to USART1_TX
- PA14 connected to USART1_RX
- PA15 connected to USART1_TX
- PA16 connected to USART1_RX
- PA17 connected to USART1_TX
- PA18 connected to USART1_RX
- PA19 connected to USART1_TX
- PA20 connected to USART1_RX
- PA21 connected to USART1_TX
- PA22 connected to USART1_RX
- PA23 connected to USART1_TX
- PA24 connected to USART1_RX
- PA25 connected to USART1_TX
- PA26 connected to USART1_RX
- PA27 connected to USART1_TX
- PA28 connected to USART1_RX
- PA29 connected to USART1_TX
- PA30 connected to USART1_RX
- PA31 connected to USART1_TX
- PA32 connected to USART1_RX
- PA33 connected to USART1_TX
- PA34 connected to USART1_RX
- PA35 connected to USART1_TX
- PA36 connected to USART1_RX
- PA37 connected to USART1_TX
- PA38 connected to USART1_RX
- PA39 connected to USART1_TX
- PA40 connected to USART1_RX
- PA41 connected to USART1_TX
- PA42 connected to USART1_RX
- PA43 connected to USART1_TX
- PA44 connected to USART1_RX
- PA45 connected to USART1_TX
- PA46 connected to USART1_RX
- PA47 connected to USART1_TX
- PA48 connected to USART1_RX
- PA49 connected to USART1_TX
- PA50 connected to USART1_RX
- PA51 connected to USART1_TX
- PA52 connected to USART1_RX
- PA53 connected to USART1_TX
- PA54 connected to USART1_RX
- PA55 connected to USART1_TX
- PA56 connected to USART1_RX
- PA57 connected to USART1_TX
- PA58 connected to USART1_RX
- PA59 connected to USART1_TX
- PA60 connected to USART1_RX
- PA61 connected to USART1_TX
- PA62 connected to USART1_RX
- PA63 connected to USART1_TX
- PA64 connected to USART1_RX
- PA65 connected to USART1_TX
- PA66 connected to USART1_RX
- PA67 connected to USART1_TX
- PA68 connected to USART1_RX
- PA69 connected to USART1_TX
- PA70 connected to USART1_RX
- PA71 connected to USART1_TX
- PA72 connected to USART1_RX
- PA73 connected to USART1_TX
- PA74 connected to USART1_RX
- PA75 connected to USART1_TX
- PA76 connected to USART1_RX
- PA77 connected to USART1_TX
- PA78 connected to USART1_RX
- PA79 connected to USART1_TX
- PA80 connected to USART1_RX
- PA81 connected to USART1_TX
- PA82 connected to USART1_RX
- PA83 connected to USART1_TX
- PA84 connected to USART1_RX
- PA85 connected to USART1_TX
- PA86 connected to USART1_RX
- PA87 connected to USART1_TX
- PA88 connected to USART1_RX
- PA89 connected to USART1_TX
- PA90 connected to USART1_RX
- PA91 connected to USART1_TX
- PA92 connected to USART1_RX
- PA93 connected to USART1_TX
- PA94 connected to USART1_RX
- PA95 connected to USART1_TX
- PA96 connected to USART1_RX
- PA97 connected to USART1_TX
- PA98 connected to USART1_RX
- PA99 connected to USART1_TX
- PA100 connected to USART1_RX

تنظیمات GPIO در STM32

تنظیمات اولیه پورت سریال UART

سپس به بخش Pinout & Configuration بازگشته و از بخش Connectivity گزینه UART1 را انتخاب نمایید. در ابتدا Mode را به حالت Asynchronous تغییر داده و در بخش Parameters Settings باودریت مورد نظر و تنظیمات دلخواه (مطابق با نیاز پروژه) را انجام می‌دهیم. چون نیاز به دریافت و ارسال دیتا داریم حالت را بروی Receive and Transmit تنظیم شده است. سپس از همین بخش وارد تب NVIC Setting شده و وقفه سریال را فعال می‌کنیم.



USART1 Mode and Configuration

Mode: Asynchronous
Hardware Flow Control (RS232): Disable

Configuration

NVIC interrupt Table	Enabled	Preemption Priority	Sub Priority
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

STM32F103C8Tx LQFP48

PA9: USART1_TX
PA10: USART1_RX

فعال سازی وقفه سریال UART در STM32 و محیط Cube

برنامه دریافت دیتا از پورت سریال به روش وقفه

در برنامه زیر قبل از (while 1) بایستی دریافت دیتا از سریال فراخوانی شده و نهایتاً در انتهای زیر برنامه تابع وقفه نیز مجدداً باید آن را وارد کرد تا دریافت های بعدی انجام پذیرد. در قسمت دریافت وقفه سریال flag را یک کرده تا در ادامه برنامه با بررسی آن نسبت به تغییر وضعیت LED اقدام شود. نهایتاً آرایه ذخیره دیتا را خالی کرده تا دیتای جدید جایگزین و اجرای شرط if بدرستی انجام پذیرد.

```
#include "main.h"  
#include <string.h>  
  
UART_HandleTypeDef huart1;  
  
void SystemClock_Config(void);  
static void MX_GPIO_Init(void);
```



```
static void MX_USART1_UART_Init(void);

uint8_t Received_data[10];
char Received_flag = 0;

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    Received_flag = 1;
    HAL_UART_Receive_IT(&huart1, Received_data, 7);
}

int main(void)
{
    HAL_Init();

    SystemClock_Config();

    MX_GPIO_Init();
    MX_USART1_UART_Init();

    HAL_UART_Receive_IT(&huart1, Received_data, 7);

    while (1)
    {
        if (Received_flag == 1)
        {
            Received_flag = 0;
            if (strcmp((const char*)Received_data, "toggle") == 0 ||
                strcmp((const char*)Received_data, "toggle\r") == 0)
            {
                HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
                memset(Received_data, 0, 10);
            }
        }
    }
}
```




```
    }  
  }  
}  
  
void SystemClock_Config(void)  
{  
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};  
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};  
  
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;  
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;  
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;  
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)  
  {  
    Error_Handler();  
  }  
  
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK  
    |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;  
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSE;  
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;  
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;  
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;  
  
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)  
  {  
    Error_Handler();  
  }  
}
```



```
static void MX_USART1_UART_Init(void)
{
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 9600;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK)
    {
        Error_Handler();
    }
}

static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);

    GPIO_InitStructure.Pin = LED_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStructure);
}
```



```
void Error_Handler(void)
{
    __disable_irq();
    while (1)
    {
    }
}

#ifdef USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line
    number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
    */
    /* USER CODE END 6 */
}

#endif
#endif
```

نتیجه گیری

در این قسمت از مطالب آموزش ارتباط سریال STM32 به معرفی و بررسی روش non-blocking mode بصورت وقفه سریال پرداخته شد. همانطور که بیان شد اگر دیتا مورد نظر طولانی باشند باعث بروز وقفه های زیادی خواهد شد. لذا روش وقفه برای این مورد مناسب نبوده و باید از روش DMA برای دیتا های طولانی یا حجیم استفاده نمود.

امیدوارم از این مطلب کمال بهره را برده باشید. در صورت داشتن هرگونه نظر یا سوال درباره این مطلب یا تجربه مشابه اون رو در انتهای همین صفحه در قسمت دیدگاه ها قرار بدید. در کوتاه ترین زمان ممکن به اون ها پاسخ



خواهم داد. اگر این مطلب براتون فید بود، اون رو به اشتراک بگذارید تا سایر دوستان هم بتوانند استفاده کنند. همینطور میتونید این مطلب را توی اینستاگرام با هشتگ #microelecom به اشتراک بگذارید و **پیج مایکروالکام** (@microelecom) رو هم منشن کنید.