**CHUYÊN ĐỀ THIẾT KẾ HỆ THỐNG NHÚNG 1**

**BÀI THỰC HÀNH 2: LẬP TRÌNH UART và RTOS TRÊN KIT OPEN405R-C PACKAGE A, STM32F4 DEVELOPMENT BOARD**

# Mục tiêu

* Giúp sinh viên làm quen với chuẩn giao tiếp UART trên vi điều khiển STM32F405, thực hiện việc giao tiếp giữa vi điều khiển STM32F405 và máy tính thông qua UART
* Giúp sinh viên làm quen với hệ điều hành thời gian thực FreeRTOS để có thể quản lý các tác vụ đòi hỏi đáp ứng theo thời gian thực

# Chuẩn bị trước

* Tìm hiểu lý thuyết cơ bản về chuẩn giao tiếp UART [1] [2]
* Tìm hiểu các lý thuyết liên quan đến hệ điều hành thời gian thực [3] [4]
* Cài đặt phần mềm Terminal [5]

# Nội dung thực hành

## Lập trình giao tiếp UART trên KIT Open405R-C Package A, STM32F4 Development Board

Sinh viên viết chương trình để thực hiện giao tiếp giữa KIT Open405R-C Package A, STM32F4 Development Board và máy tính.

## Lập trình quản lý các tác vụ trên FreeRTOS chạy trên KIT Open405R-C Package A, STM32F4 Development Board

Sinh viên làm quen với việc tạo và quản lý các tác vụ thời gian thực trên vi điều khiển với việc sử dụng Hệ điều hành thời gian thực FreeRTOS.

# Hướng dẫn thực hành

## Diagram, schematic Description automatically generatedLập trình giao tiếp UART trên KIT Open405R-C Package A, STM32F4 Development Board

Sinh viên thực hiện cấu hình cho USART1 tại thẻ Connectivity dựa theo sơ đồ nguyên lý mà nhà sản xuất cung cấp TX:PA9, RX: PA10.

Sinh viên chọn Mode Asynchronous và cài đặt các thông số Baud Rate, Word Length, Parity và Stop Bits cho phù hợp với ứng dụng như trong **Hình 2**. Trong trường hợp muốn sử dụng interrupt cho UART thì sinh viên chọn cấu hình ở thẻ NVIC Settings.

**Hình 1 Sơ đồ chân UART 1**

Để thực hiện việc gửi dữ liệu thông qua UART từ vi điều khiển, sinh viên có thể sử dụng một trong các hàm HAL\_GPIO\_Transmit / HAL\_GPIO\_Transmit\_IT / HAL\_GPIO\_Transmit\_DMA.

Ngoài ra, sinh viên có thể retarget để chuyển UART thành output mặc định của hàm printf thông qua đoạn prototype sau (*Lưu ý: Sinh viên phải include thư viện stdio.h để có thể sử dụng hàm printf*):

**#ifdef** \_\_GNUC\_\_

/\* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf

set to 'Yes') calls \_\_io\_putchar() \*/

**#define** **PUTCHAR\_PROTOTYPE** **int** \_\_io\_putchar(**int** ch)

**#else**

**#define** **PUTCHAR\_PROTOTYPE** **int** fputc(**int** ch, FILE \*f)

**#endif** /\* \_\_GNUC\_\_ \*/

/\*\*

\* @brief Retargets the C library printf function to the USART.

\* @param None

\* @retval None

\*/

PUTCHAR\_PROTOTYPE

{

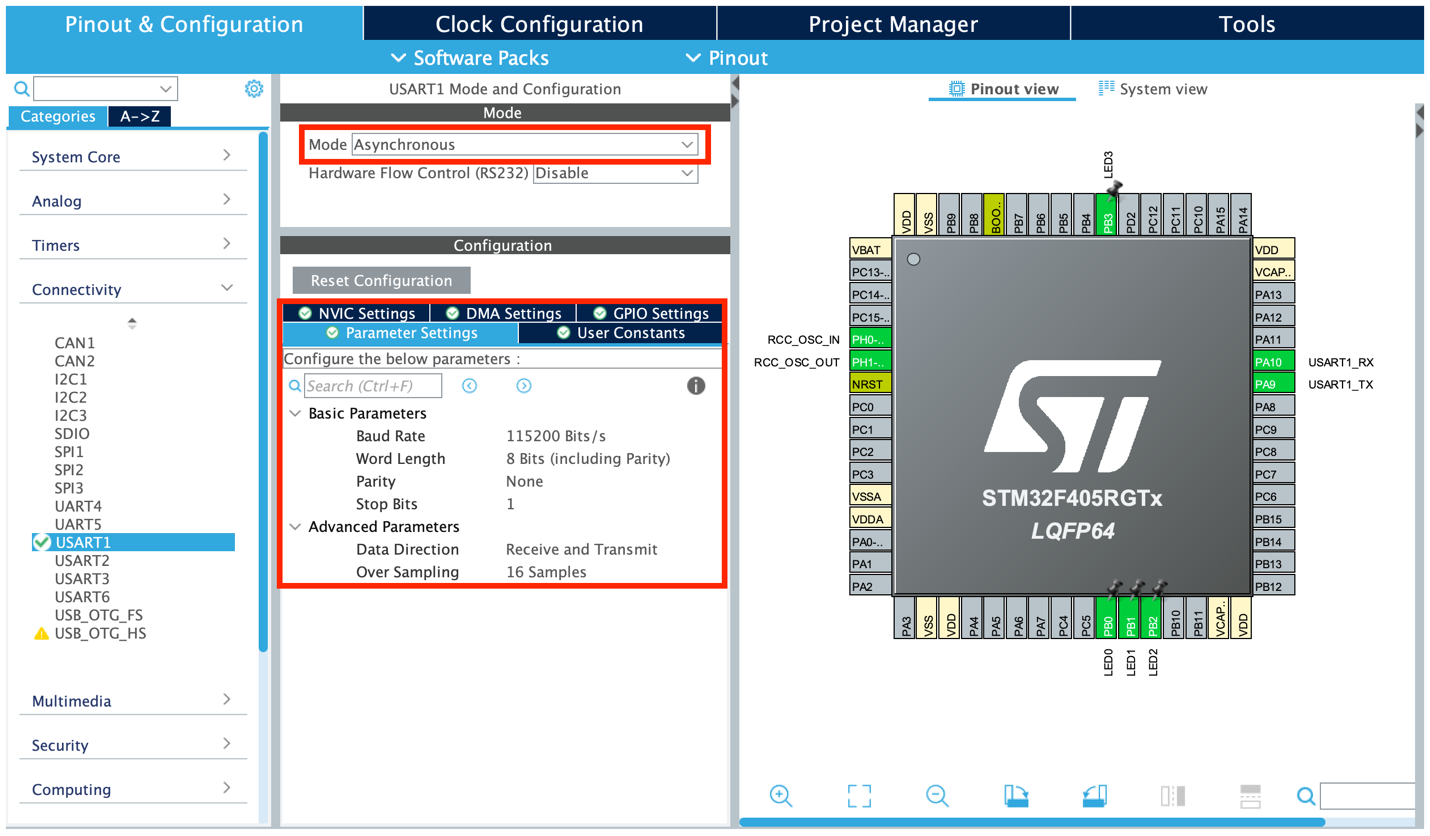
/\* Place your implementation of fputc here \*/

/\* e.g. write a character to the USART \*/

**HAL\_UART\_Transmit**(&huart3, (**uint8\_t**\*) &ch, 1, 100);

**return** ch;

}



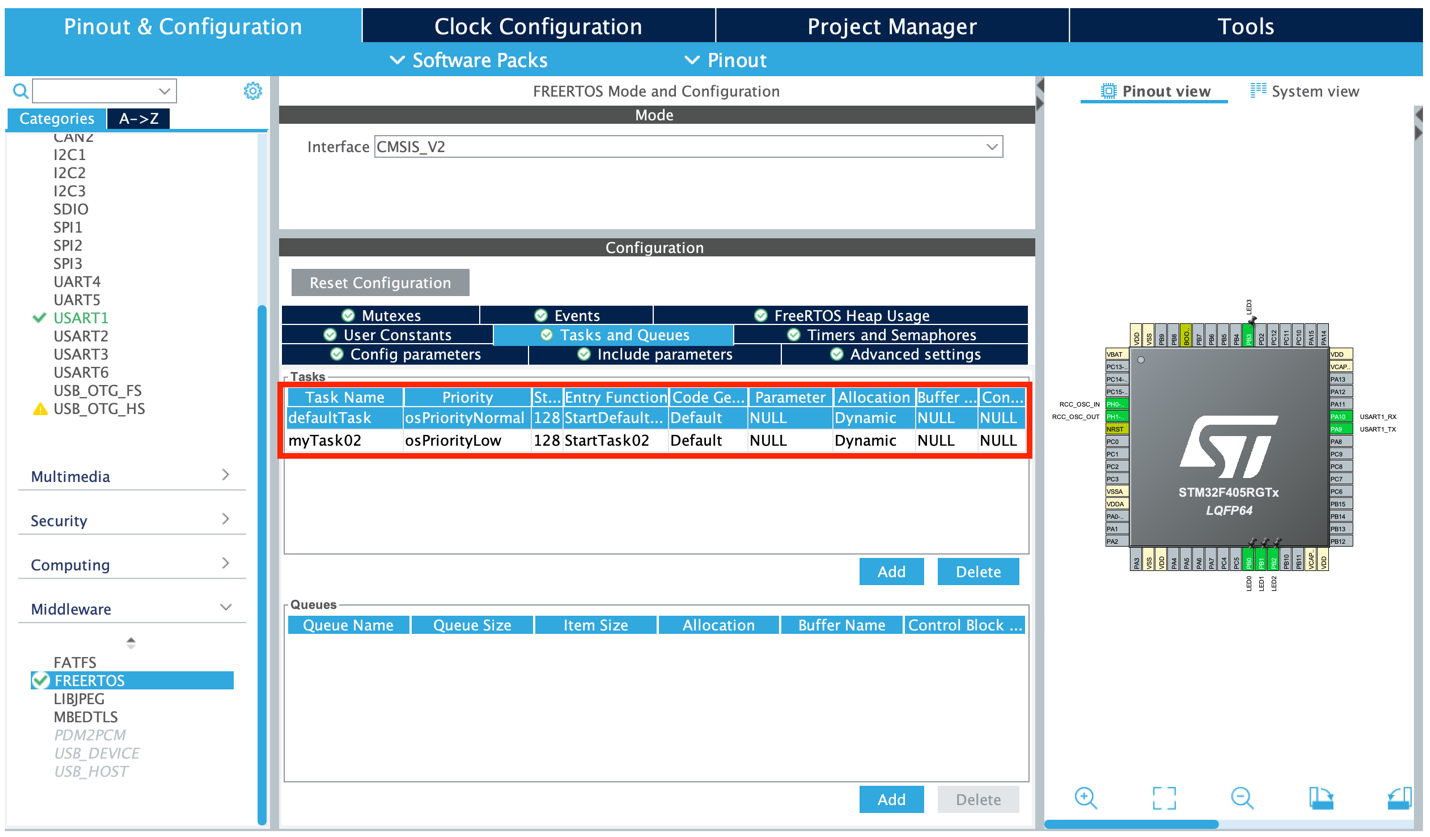
**Hình 2 Cấu hình các thông số cho UART 1**

Sau khi hoàn thành chương trình, sinh viên sử dụng phần mềm terminal để kết nối với vi điều khiển thông qua cổng COM Port trên máy tính. Cấu hình thông số cho cổng COM Port phù hợp với thông số mà sinh viên đã cài đặt cho cổng UART trên vi điều khiển.

## Lập trình quản lý các tác vụ trên FreeRTOS chạy trên KIT Open405R-C Package A, STM32F4 Development Board

Hệ điều hành thời gian thực là một hệ điều hành có kích thước nhỏ gọn thường được dùng cho các vi điều khiển để quản lý các tác vụ trong ứng dụng để có thể đáp ứng được các yêu cầu nghiêm ngặt theo thời gian thực với số lượng tài nguyên hữu hạn và rất hạn chế. Hệ điều hành thời gian thực là một phần của chương trình, trong đó, hệ điều hành thời gian thực sẽ giải quyết việc điều phối các tác vụ, lập lịch và phân mức ưu tiên cho các tác vụ, quản lý các thông điệp trao đổi giữa các tác vụ. Trong ứng dụng STM32CubeIDE, nhà phát hành đã tích hợp sẵn 1 hệ điều hành thời gian thực FreeRTOS để các nhà lập trình có thể tích hợp lên các ứng dụng của các dòng vi điều khiển STM32.

Để cấu hình sử dụng hệ điều hành thời gian thực, sinh viên chọn thẻ Middleware và chọn FREERTOS. Chọn Interface CMSIS\_V2 là phiên bản mới nhất của FreeRTOS. Ngoài ra, sinh viên chọn thẻ Tasks and Queues để quản lý các tác vụ như . Ngoài ra, sinh viên có thể sử dụng Mutexes, Timers and Semaphores để có thể hỗ trợ việc đồng bộ và các yêu cầu khác của hệ thống.



**Hình 3 Cấu hình các tác vụ thời gian thực**

# Bài tập thực hành

Sinh viên thực hiện chương trình thời gian thực để hoàn thành các tác vụ sau:

* Tác vụ 1: Đọc các dữ liệu nhận được từ UART
* Tác vụ 2: Xuất ra dữ liệu đã nhận được từ UART, xuất ra theo chu kỳ 500ms bản tin: “Chương trình đang chạy…”
* Tác vụ 3: Xử lý dữ liệu nhận được từ UART và điều khiển các LED theo dữ liệu đã được nhận từ UART

# Báo cáo thực hành

Sinh viên viết báo cáo theo mẫu đã được cung cấp để trình bày lại các kết quả đã làm được trong bài thực hành. Giải thích các đoạn chương trình mà sinh viên đã hiện thực để giải quyết các yêu cầu của bài thực hành.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | "wiki," [Online]. Available: https://en.wikipedia.org/wiki/Universal\_asynchronous\_receiver-transmitter. [Accessed 2022]. |
| [2] | "TAPIT UART," [Online]. Available: https://tapit.vn/truyen-thong-noi-tiep-trong-lap-trinh-vi-dieu-khien-giao-tiep-uart/. [Accessed 2022]. |
| [3] | "TAPIT RTOS," [Online]. Available: https://tapit.vn/dieu-hanh-rtos/. [Accessed 2022]. |
| [4] | [Online]. Available: https://sites.google.com/site/terminalbpp/. [Accessed 2022]. |
| [5] | "Open405R-C Package A, STM32F4 Development Board," Waveshare Electronics, 2022. [Online]. Available: https://www.waveshare.com/open405r-c-package-a.htm. |
| [6] | Waveshare Electronics, [Online]. Available: https://www.waveshare.com/w/upload/f/fe/OpenX05R-C-Schematic.pdf. [Accessed 2022]. |
| [7] | STMicroelectronics, [Online]. Available: https://www.st.com/resource/en/datasheet/dm00037051.pdf. [Accessed 2022]. |