TAPiT

KHÓA HỌC
LẬP TRÌNH VI ĐIỀU KHIỂN

Giảng viên
NGUYỄN HUỲNH NHẬT THƯƠNG

LỊCH HỌC:
19h30 - 22h00 thứ 2 và thứ 6

ĐỊA ĐIỂM:
Online qua nền tảng Zoom

MODULE 6

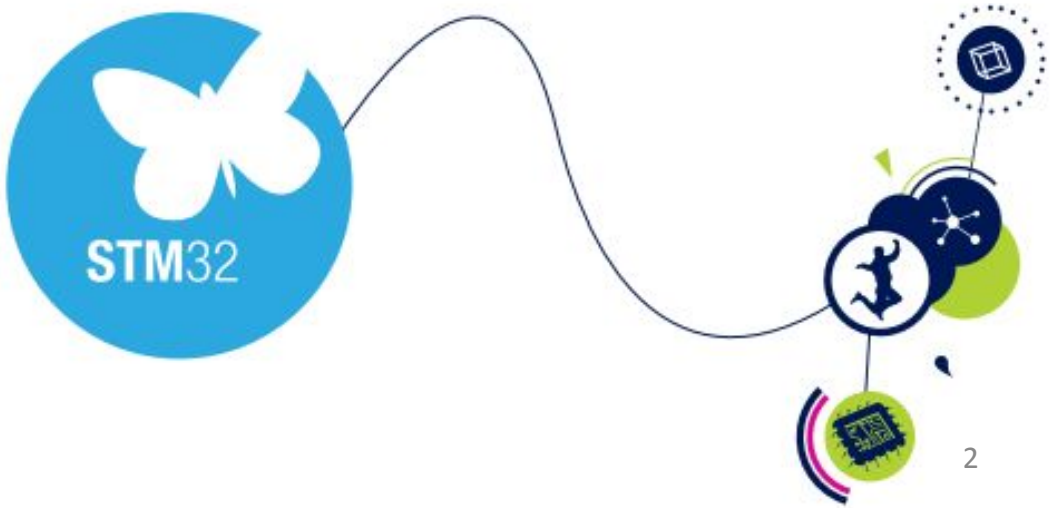
- UART - GIAO TIẾP TRUYỀN THÔNG NỐI TIẾP UART
- UART Transmit
- UART Receive

0981001119 tapitlrs@gmail.com https://tapit.vn

TAPiT

MODULE 6

- UART - GIAO TIẾP TRUYỀN THÔNG NỐI TIẾP

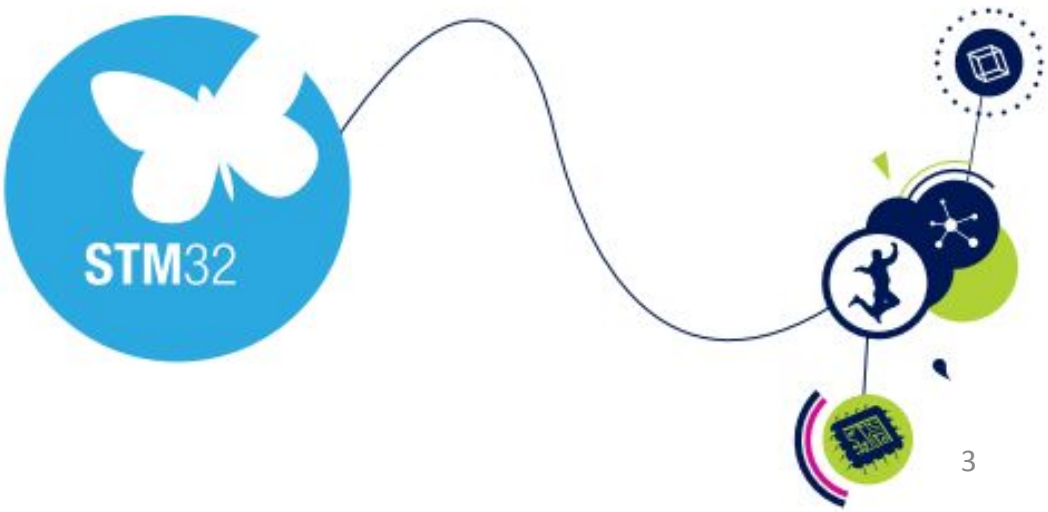


GIAO TIẾP TRUYỀN THÔNG NỐI TIẾP

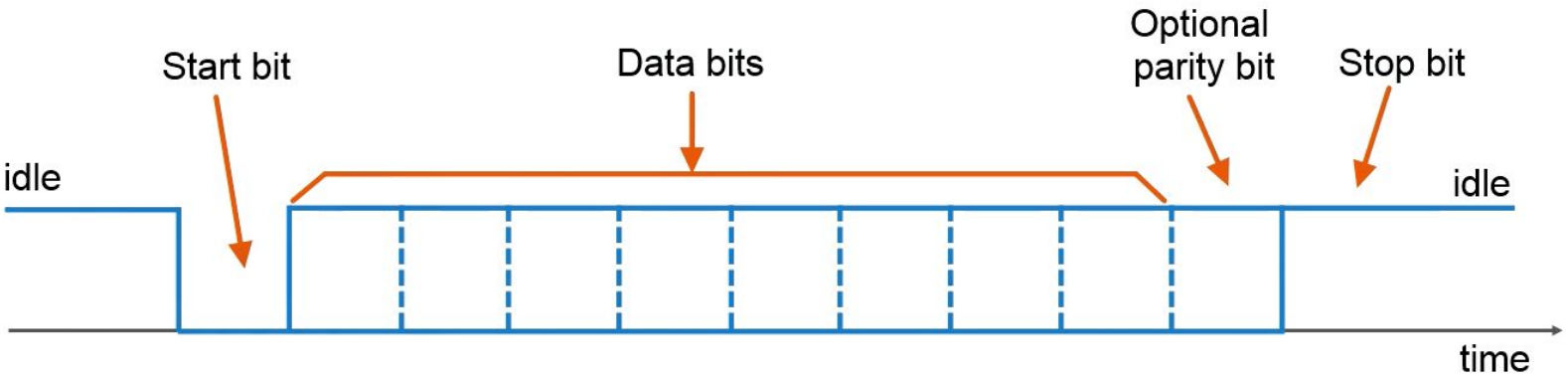
TAPiT

U A R T

Universal Asynchronous Receiver Transmitter



KHUNG TRUYỀN DỮ LIỆU (Data Frame)



- Start bit: bit bắt đầu với mức logic 0, báo hiệu truyền dữ liệu
- Data bits: các bit dữ liệu (7, 8 hoặc 9 bits)
- Parity bit: bit kiểm tra lỗi dữ liệu
- Stop bit: bit kết thúc (0.5, 1, 1.5, 2 bits) với mức logic 1, báo hiệu kết thúc khung truyền

- Biểu diễn kí tự ‘a’ dưới dạng binary
- Biểu diễn khung truyền dữ liệu khi truyền đi kí tự ‘a’ với cấu hình sau:
  - 1 bit start
  - 8 bits data
  - 0 bit parity
  - 1 stop bit
  - LSB first (bit

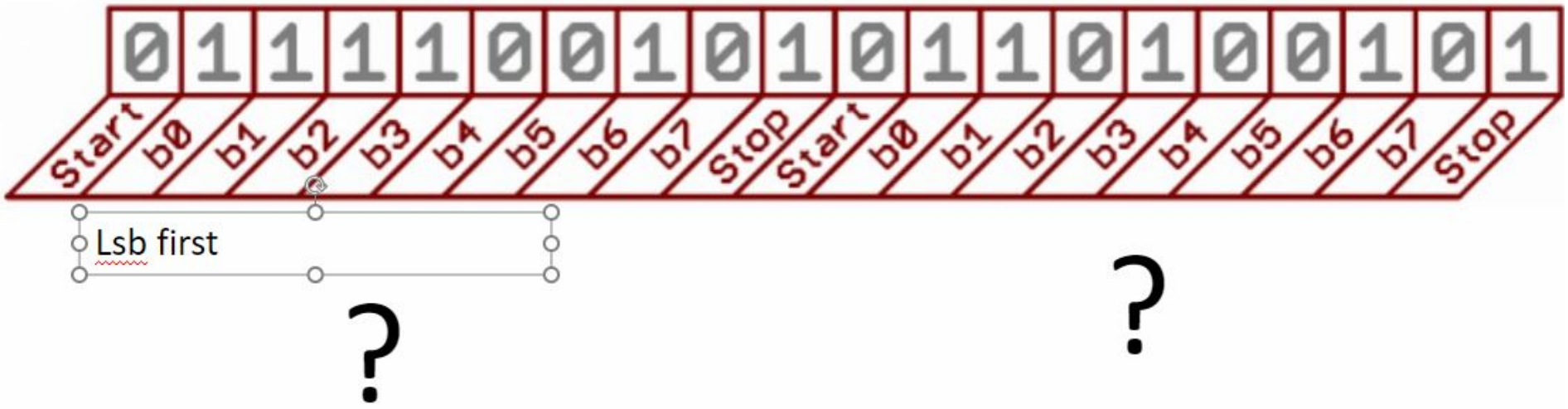
TỐC ĐỘ BAUD (BAUDRATE)

- Số bit truyền được trong 1 giây. Đơn vị: bps (bits per second)
- Cần thống nhất tốc độ baud giữa hai thiết bị với nhau để thời gian truyền 1 bit và nhận 1 là giống nhau
- Một số giá trị Baudrate thường gặp: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200…
- Tốc độ baud càng cao thì tốc độ truyền/nhận dữ liệu càng nhanh

TAPiT

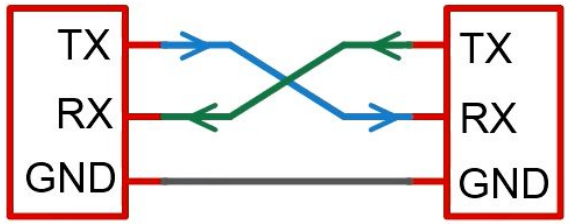
9600 8N1 (an example)

9600 baud, 8 data bits, no parity, and 1 stop bit
- is one of the more commonly used serial protocols



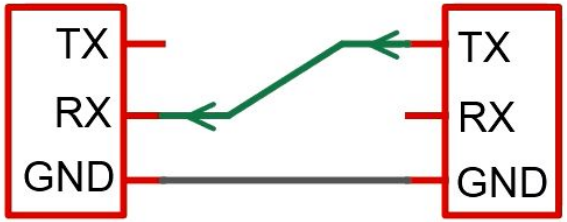
KẾT NỐI PHẦN CỨNG

- Giao tiếp hai chiều

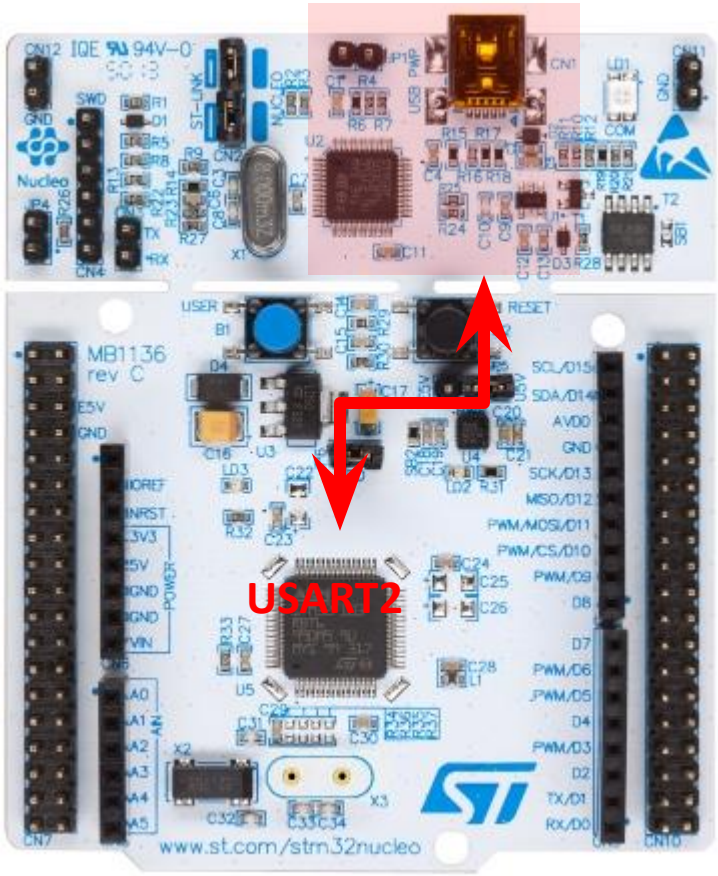


- Vi điều khiển chỉ truyền dữ liệu

- Vi điều khiển chỉ nhận dữ liệu



- TX: Transmit Data Output
- RX: Receive Data Input



https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf
(Mục 6.8)

MỘT SỐ THIẾT BỊ GIAO TIẾP UART

Màn hình Nextion

Cảm biến đo bụi không khí PMS7003

Cảm biến siêu âm MB7389 HRXL

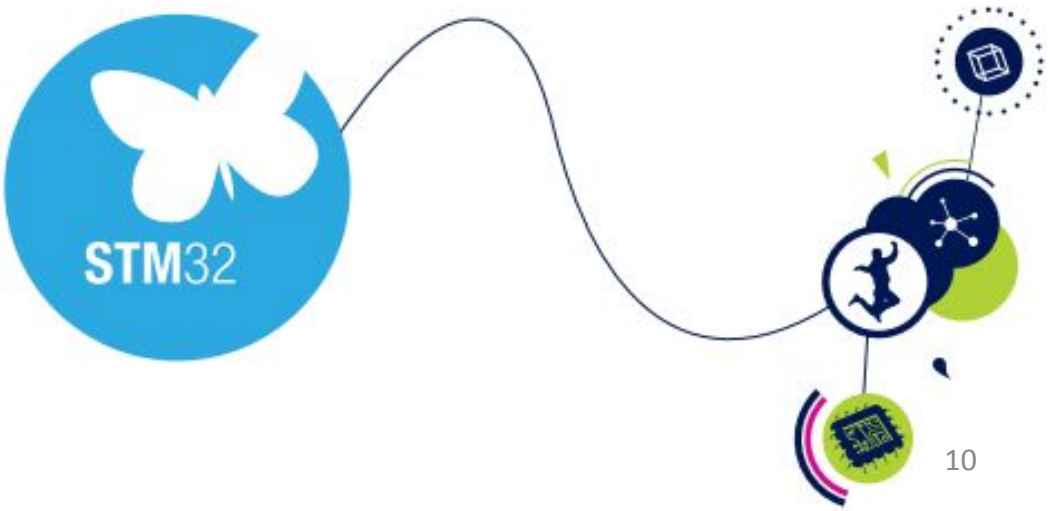
Cảm biến độ ẩm đất RS485

RS232, RS485

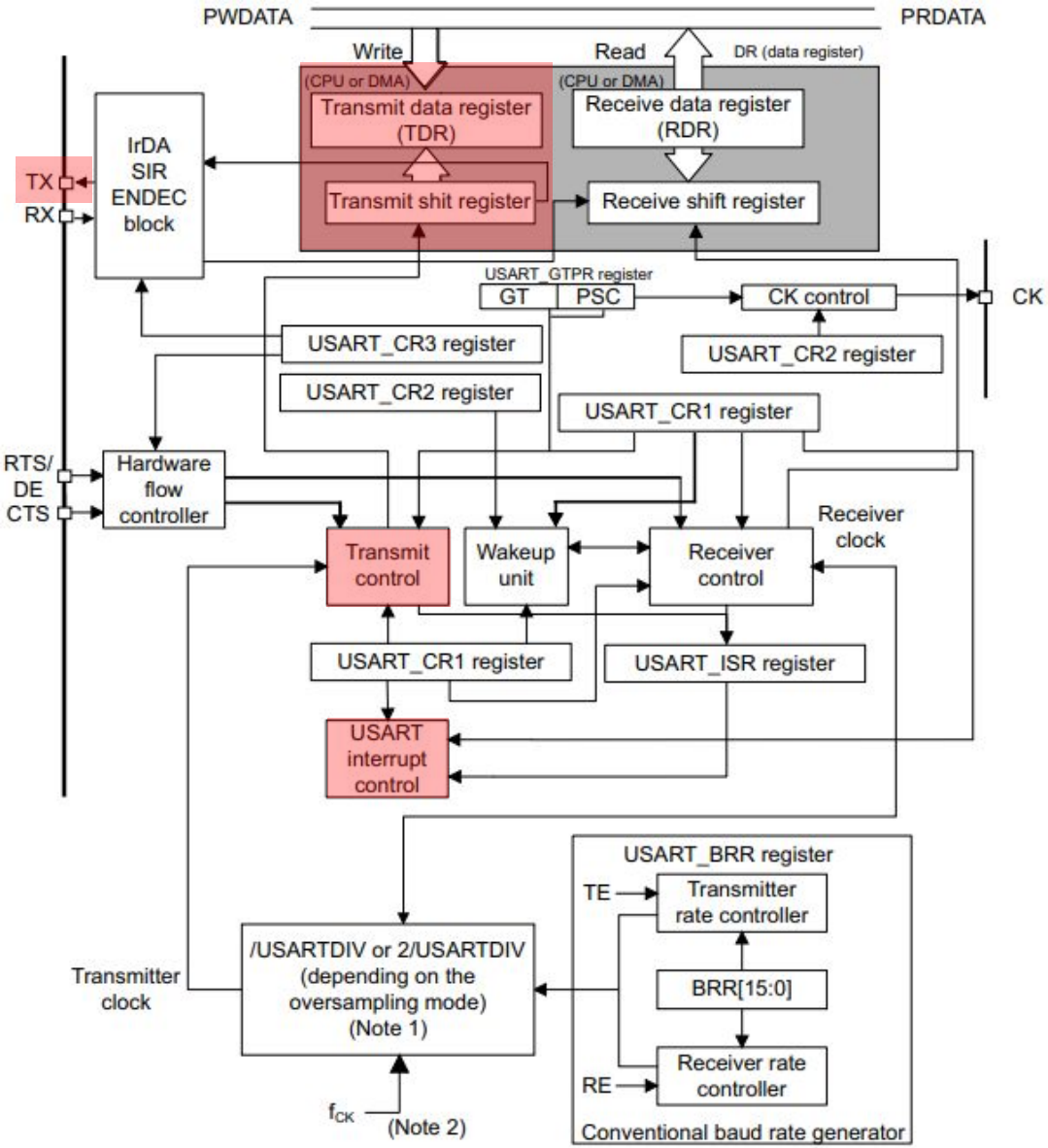
https://tapit.vn/khai-niem-ve-giao-thuc-modbus-rtu-va-ket-noi-phan-cung/

MODULE 5

- UART TRANSMIT



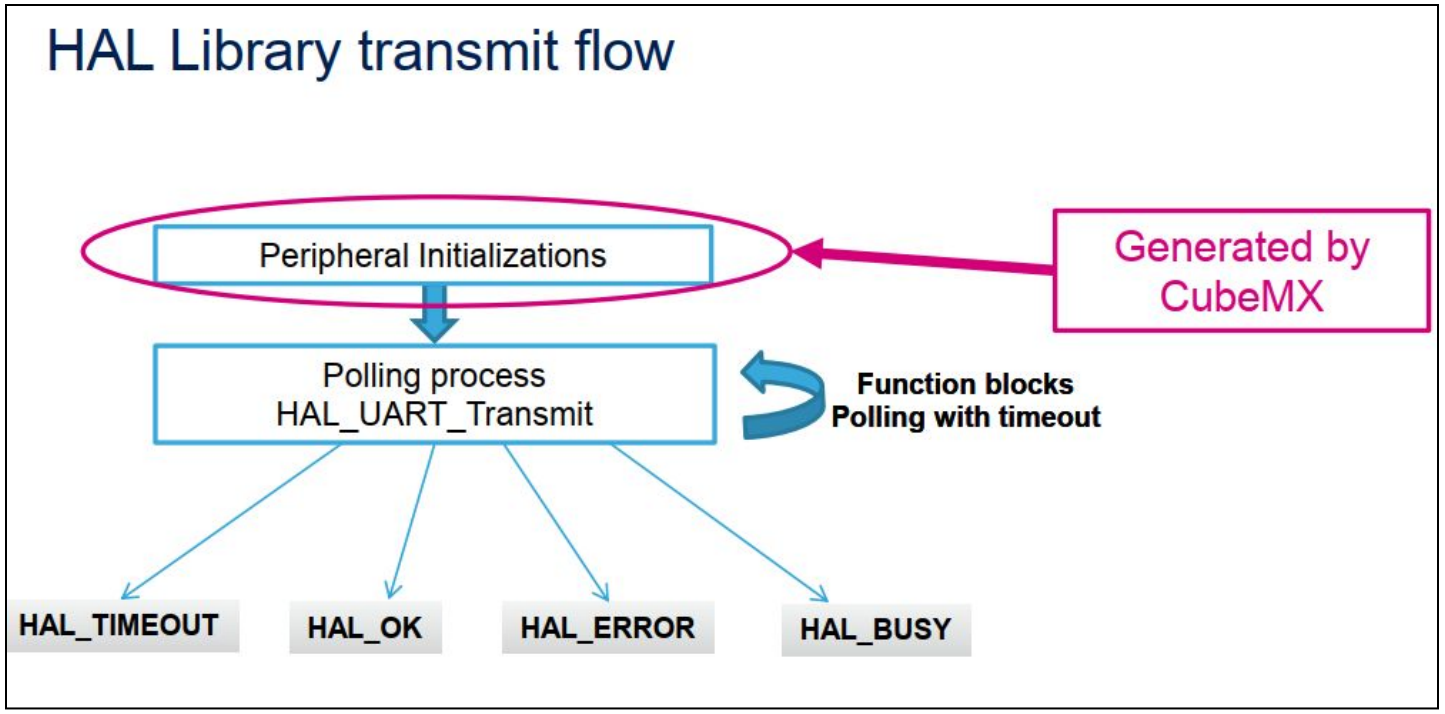
QUÁ TRÌNH TRUYỀN MỘT BYTE DỮ LIỆU



Lưu ý: Khi bộ truyền USART được kích hoạt nhưng không có hoạt động truyền diễn ra thì chân TX được duy trì mức logic cao.

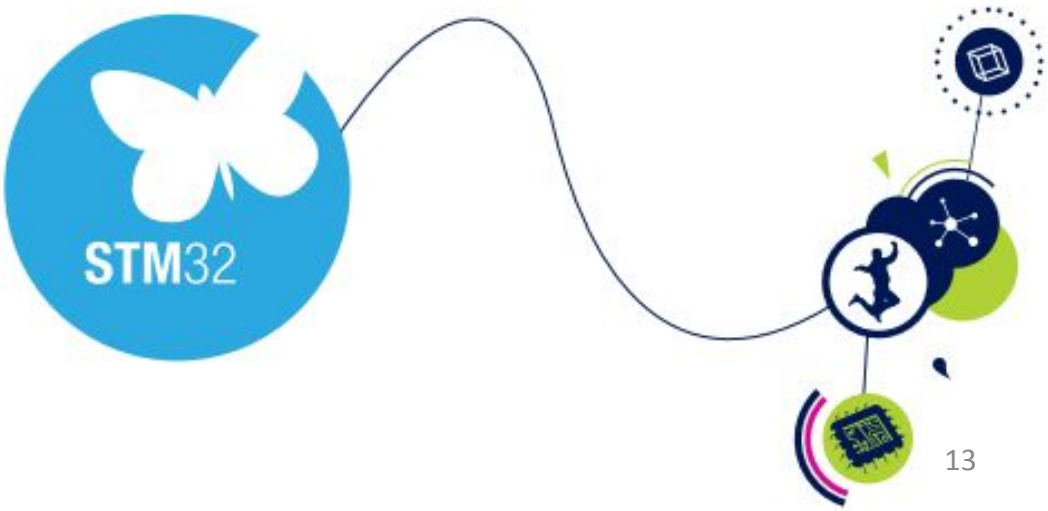
CÁC CÂU LỆNH HỖ TRỢ

- HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
- HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
- HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)

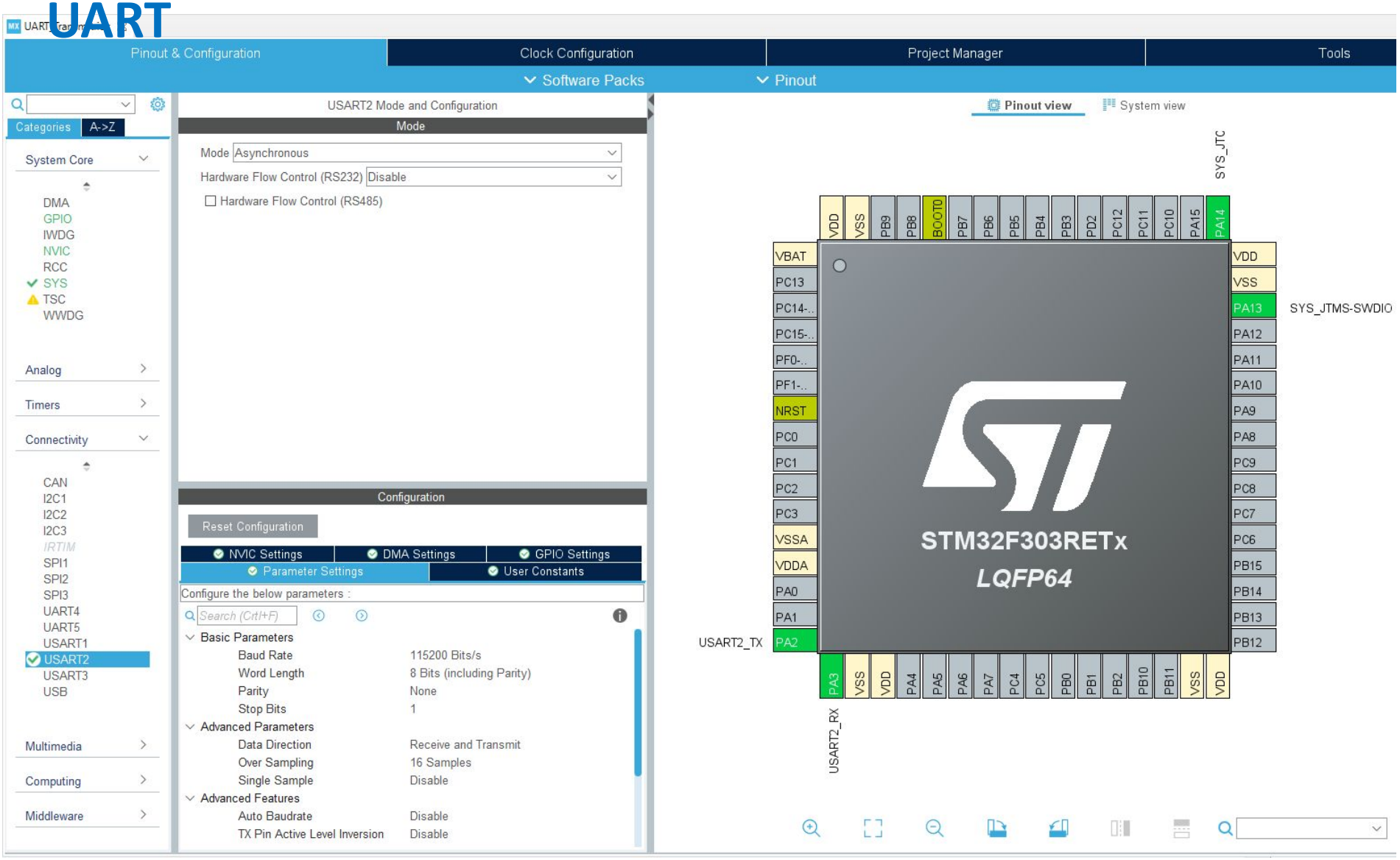


MODULE 5

- THỰC HÀNH UART TRANSMIT



CẤU HÌNH NGOẠI VI UART



VIẾT CHƯƠNG TRÌNH & NẠP CODE

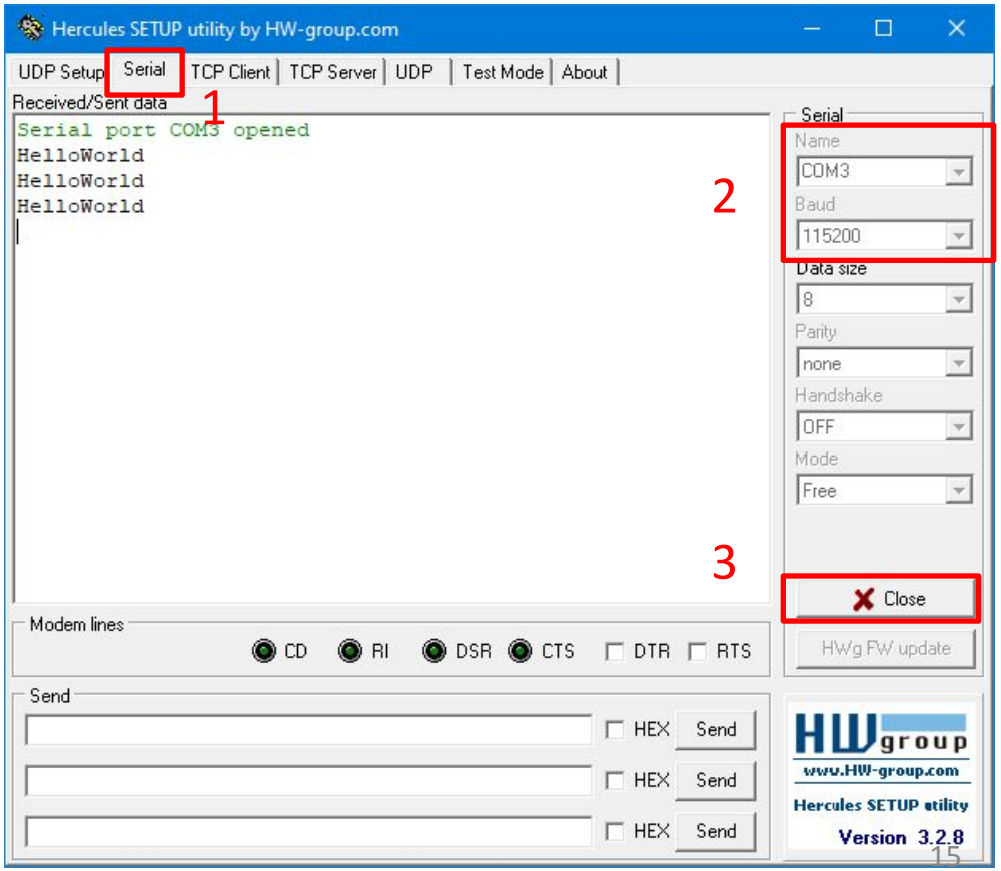
```
/* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_UART_Transmit(&huart2, (uint8_t*)"HelloWorld\n", 12, 1000);
      HAL_Delay(1000);
  }
```

PHẦN MỀM HERCULES

Download:
https://www.hw-group.com/software/hercules-setup-utility

KIỂM TRA CỔNG COM

Computer Management -> Device Manager -> Ports (COM &LPT)



TRUYỀN DỮ LIỆU NHIỀU ĐỊNH DẠNG

Truyền dữ liệu có giá trị kiểu int

```
uint16_t intnumber = 1000;
char intarray[5] = {0};
sprintf(intarray, "%d", intnumber);
HAL_UART_Transmit(&huart2, (uint8_t*)intarray, 4, 1000);
```

Truyền dữ liệu có giá trị kiểu float

```
float floatnumber = 123.4;
char floatarray[6] = {0};
sprintf(floatarray, "%.1f", floatnumber);  //Lưu ý về định dạng số lượng chữ số phần thập phân
HAL_UART_Transmit(&huart2, (uint8_t*)floatarray, 5, 1000);
```

Truyền kết hợp nhiều kiểu dữ liệu khác nhau

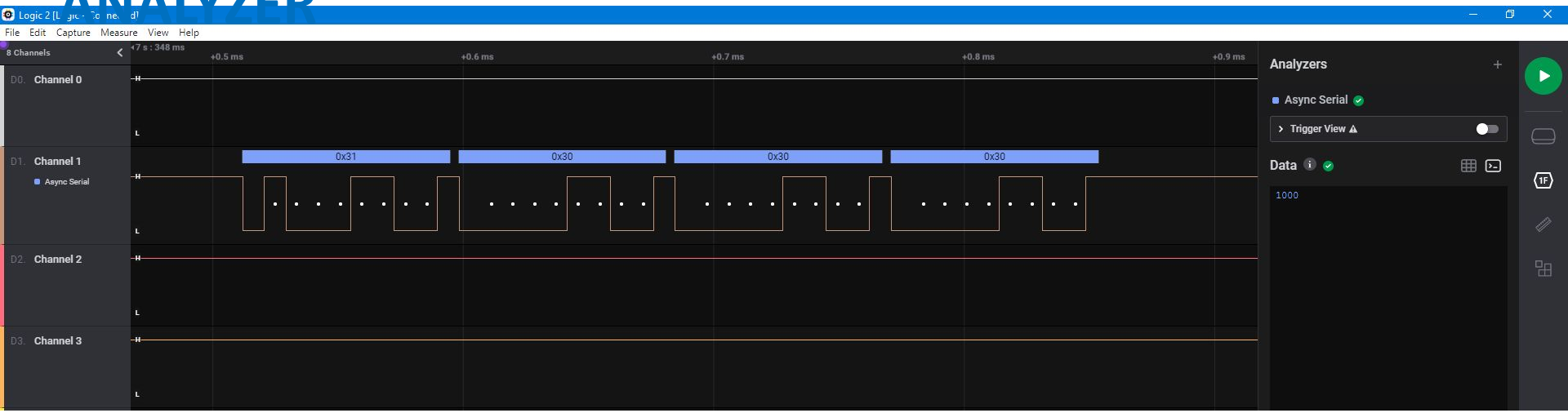
```
float tempvalue = 26.6;
uint8_t humidity = 85;
char array[20] = {0};
sprintf(array, "Nhiet do: %.1f, Do am: %d", tempvalue, humidity);
HAL_UART_Transmit(&huart2, (uint8_t*)array, 20, 1000);
```

https://tapit.vn/huong-dan-xu-ly-chuoi-trong-lap-trinh-vi-dieu-khien/

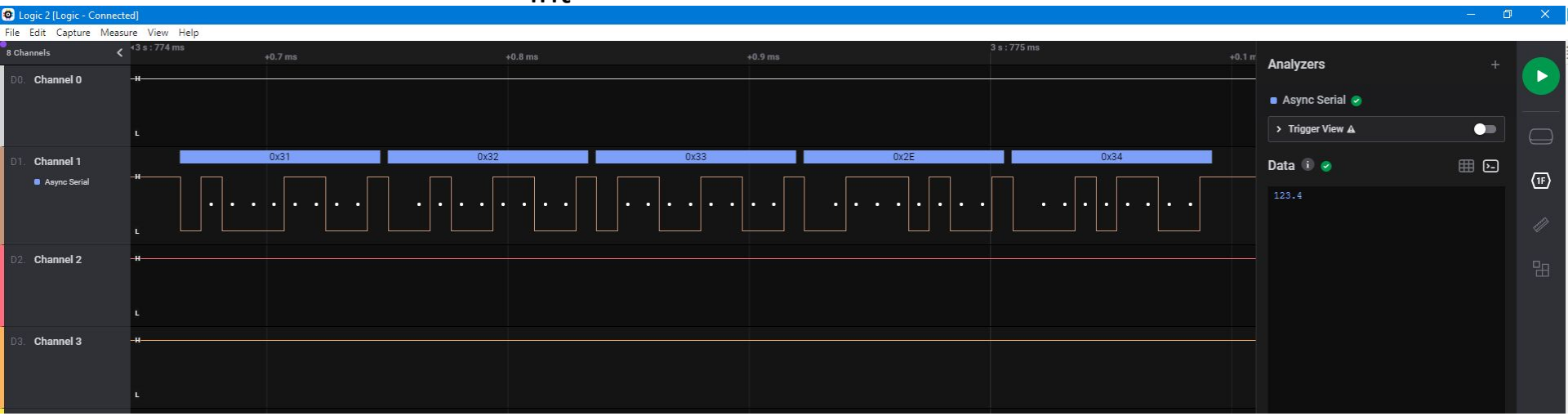
https://tapit.vn/huong-dan-xu-ly-chuoi-trong-lap-trinh-vi-dieu-khien-p2/

https://tapit.vn/huong-dan-xu-ly-chuoi-trong-lap-trinh-vi-dieu-khien-p3/

CÔNG CỤ LOGIC ANALYZER



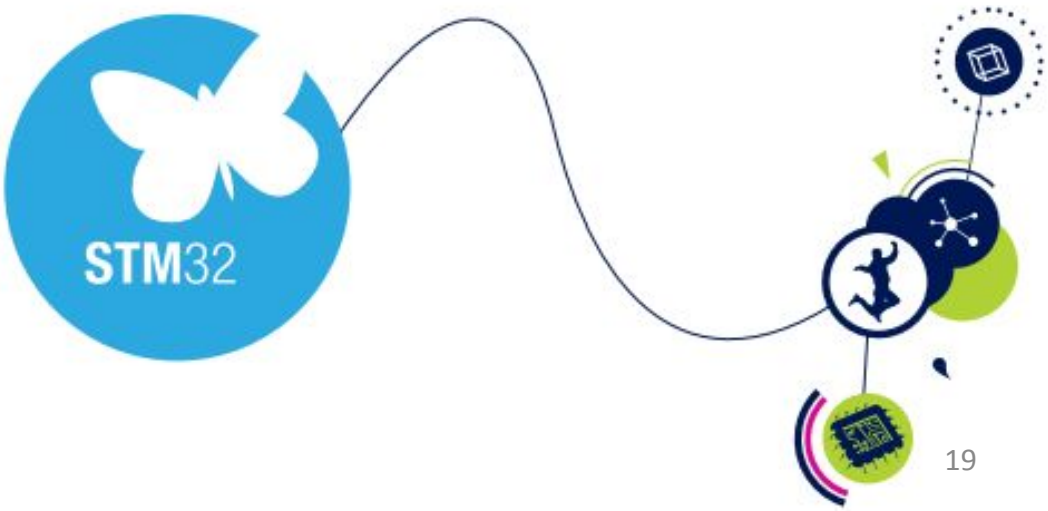
Truyền dữ liệu có giá trị kiểu int



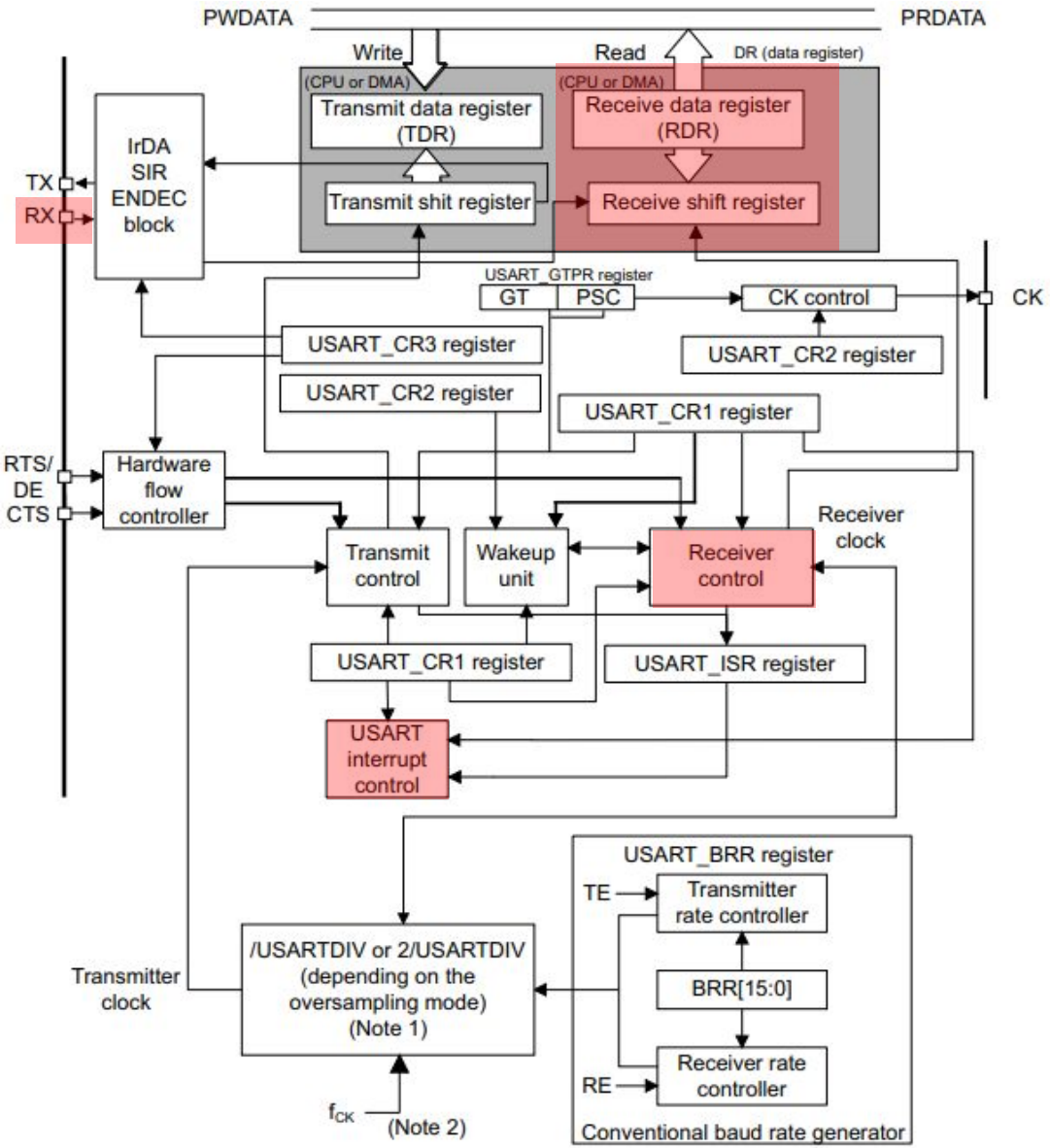
Truyền dữ liệu có giá trị kiểu float

MODULE 5

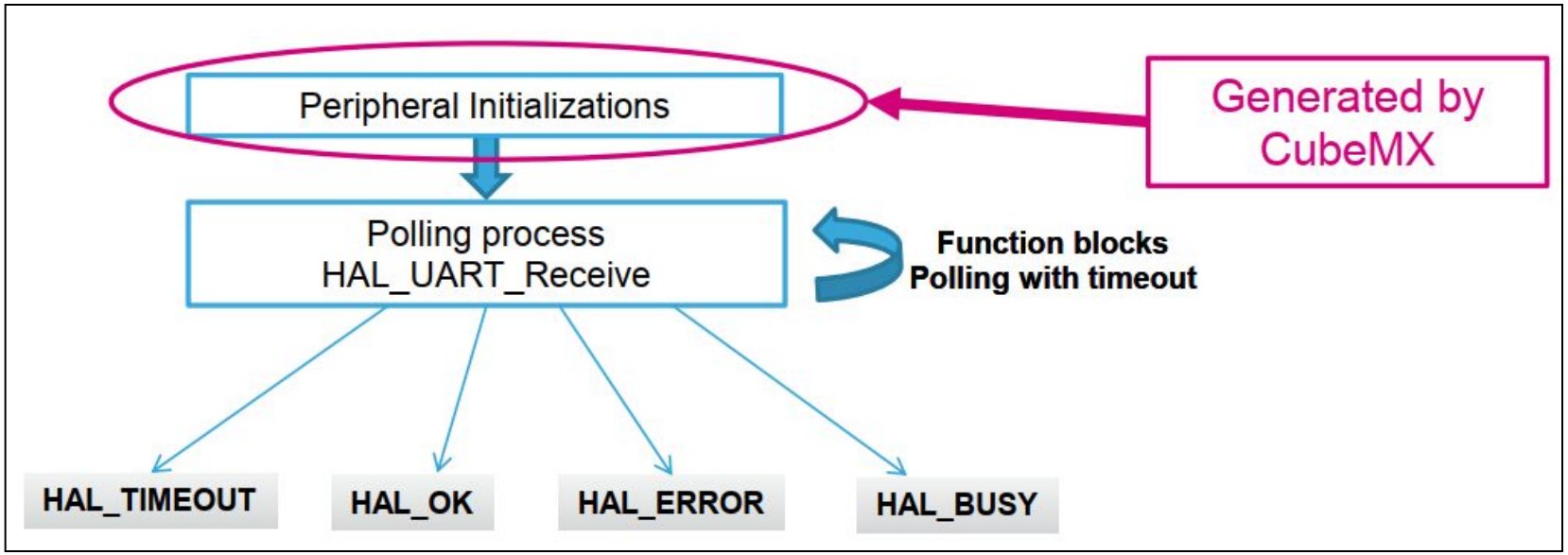
- UART RECEIVE



QUÁ TRÌNH NHẬN MỘT BYTE DỮ LIỆU



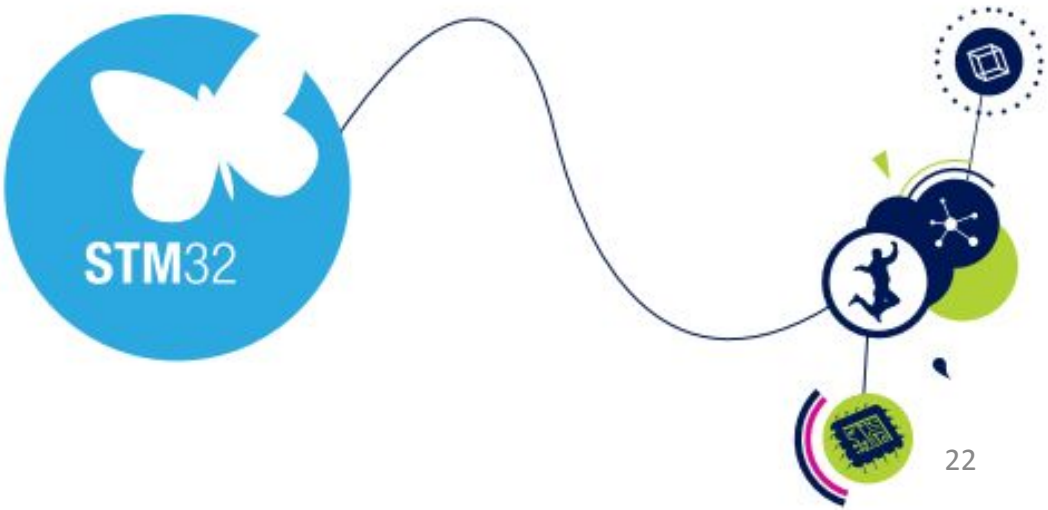
NHẬN DỮ LIỆU Ở CHẾ ĐỘ POLLING



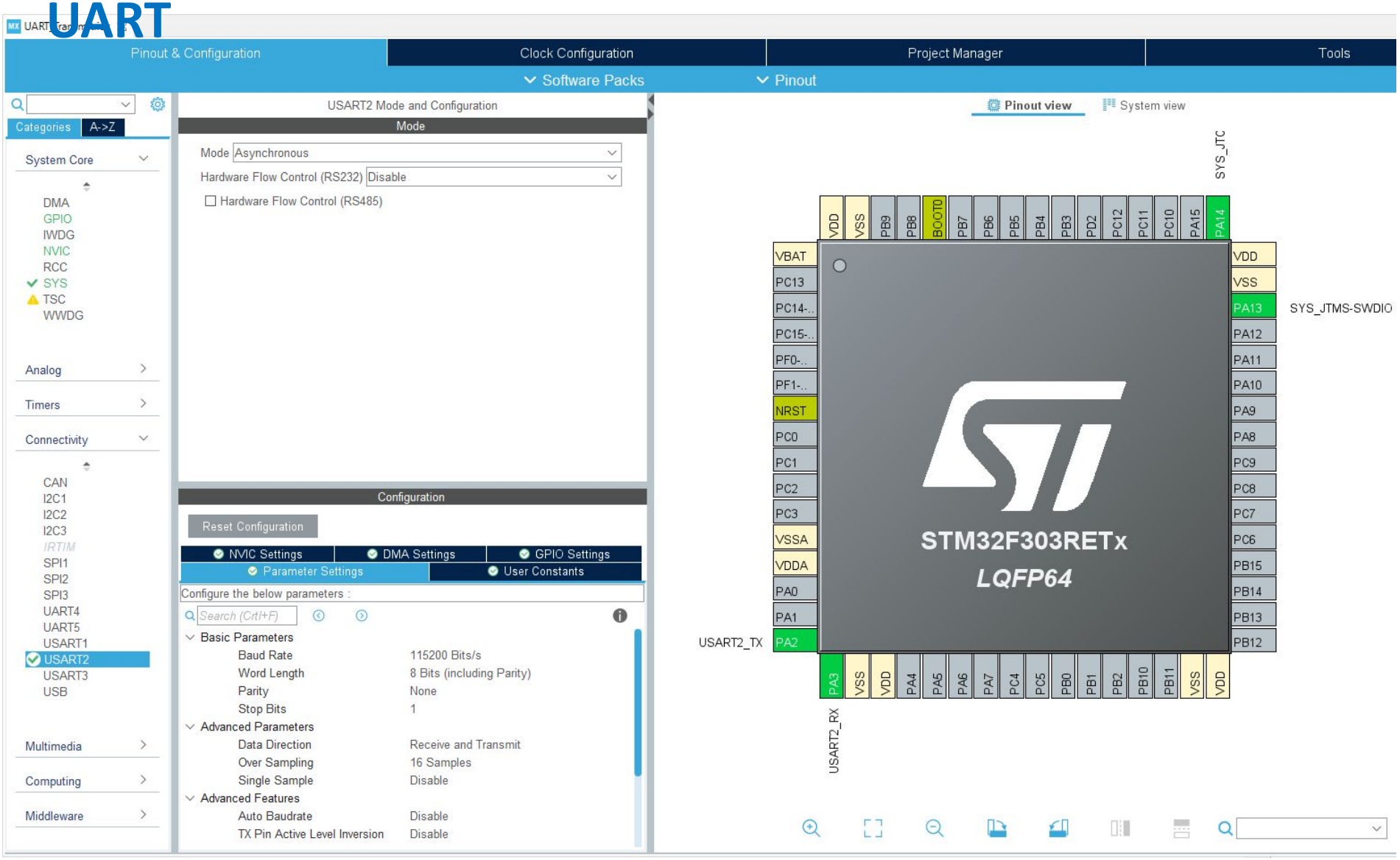
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData,
uint16_t Size, uint32_t Timeout)

MODULE 5

- THỰC HÀNH NHẬN DỮ LIỆU Ở CHẾ ĐỘ POLLING



CẤU HÌNH NGOẠI VI UART

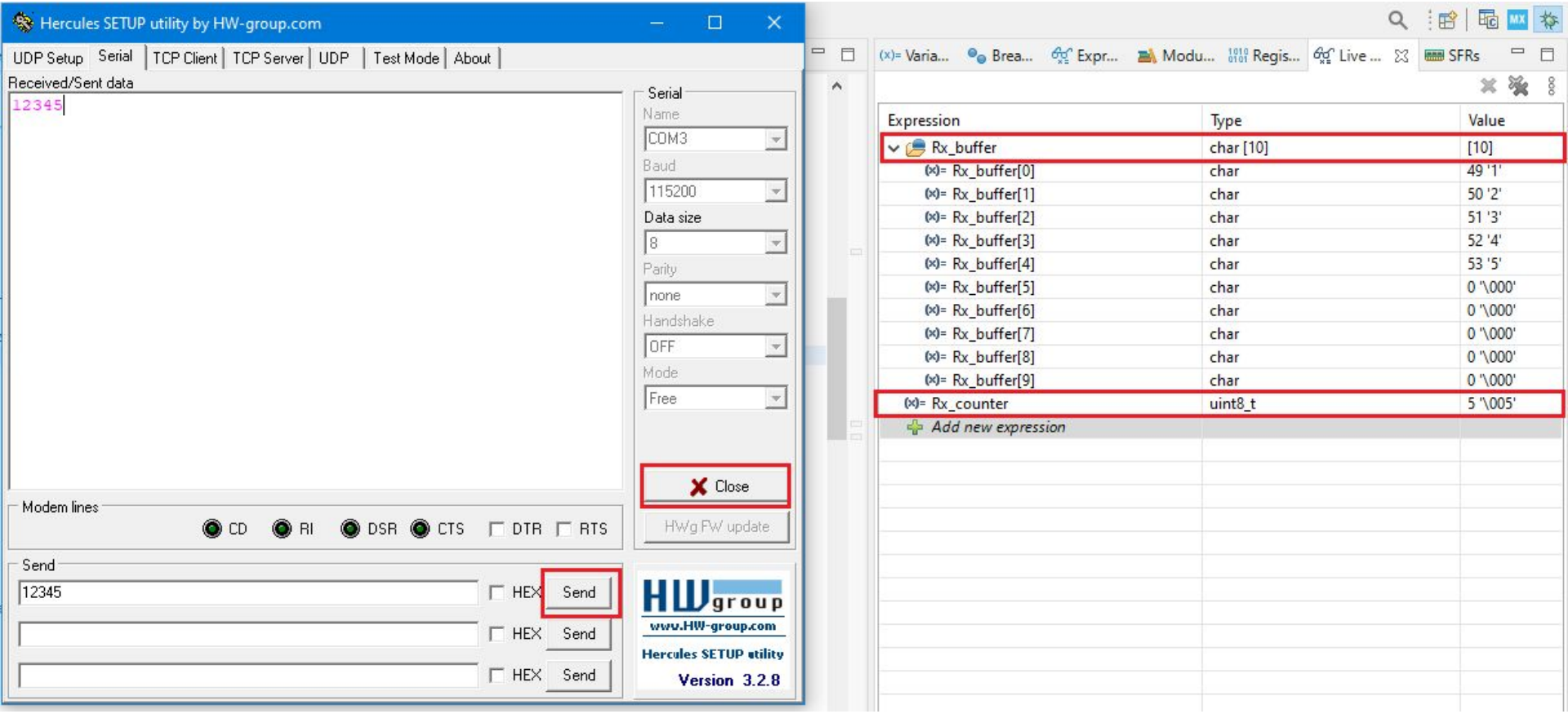


VIẾT CHƯƠNG TRÌNH & NẠP CODE

LƯU Ý: KHÔNG COPY CODE - NGUY CƠ LỖI

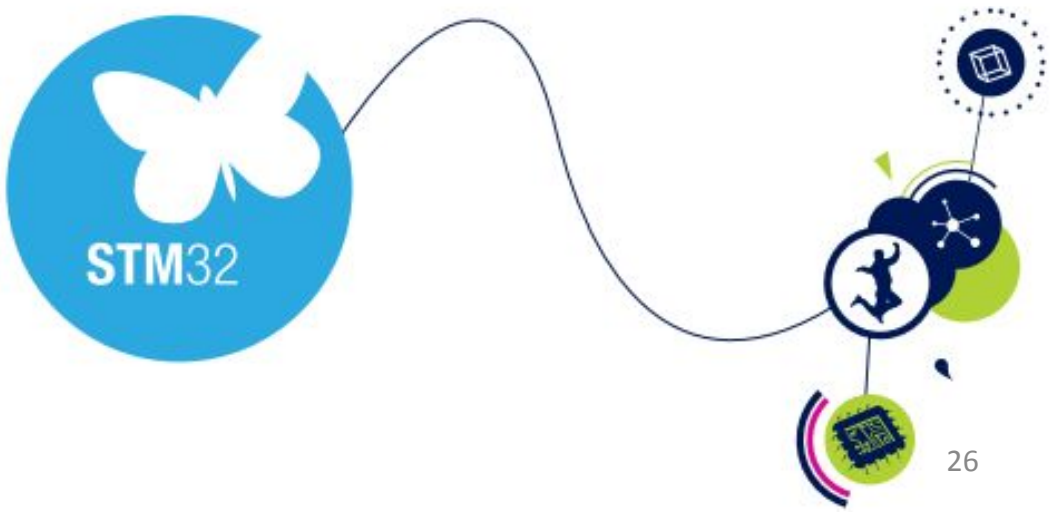
```
/* USER CODE BEGIN PV */
char Rx_byte = 0;
char Rx_buffer[10] = {0};
uint8_t Rx_counter = 0;

/* USER CODE BEGIN WHILE */
 while (1)
 {
      if(HAL_UART_Receive(&huart2, &Rx_byte, 1, 100) == HAL_OK)
      {
          if(Rx_counter >= 10) //Nếu đầy bộ đệm
          {
              for(uint8_t i = 0; i < 10; i++)  //Xóa bộ đệm
              {

                  Rx_buffer[i] = 0;
              }
             Rx_counter = 0;
          }
          Rx_buffer[Rx_counter++] = Rx_byte;
      }
 }
```

PHẦN MỀM HERCULES

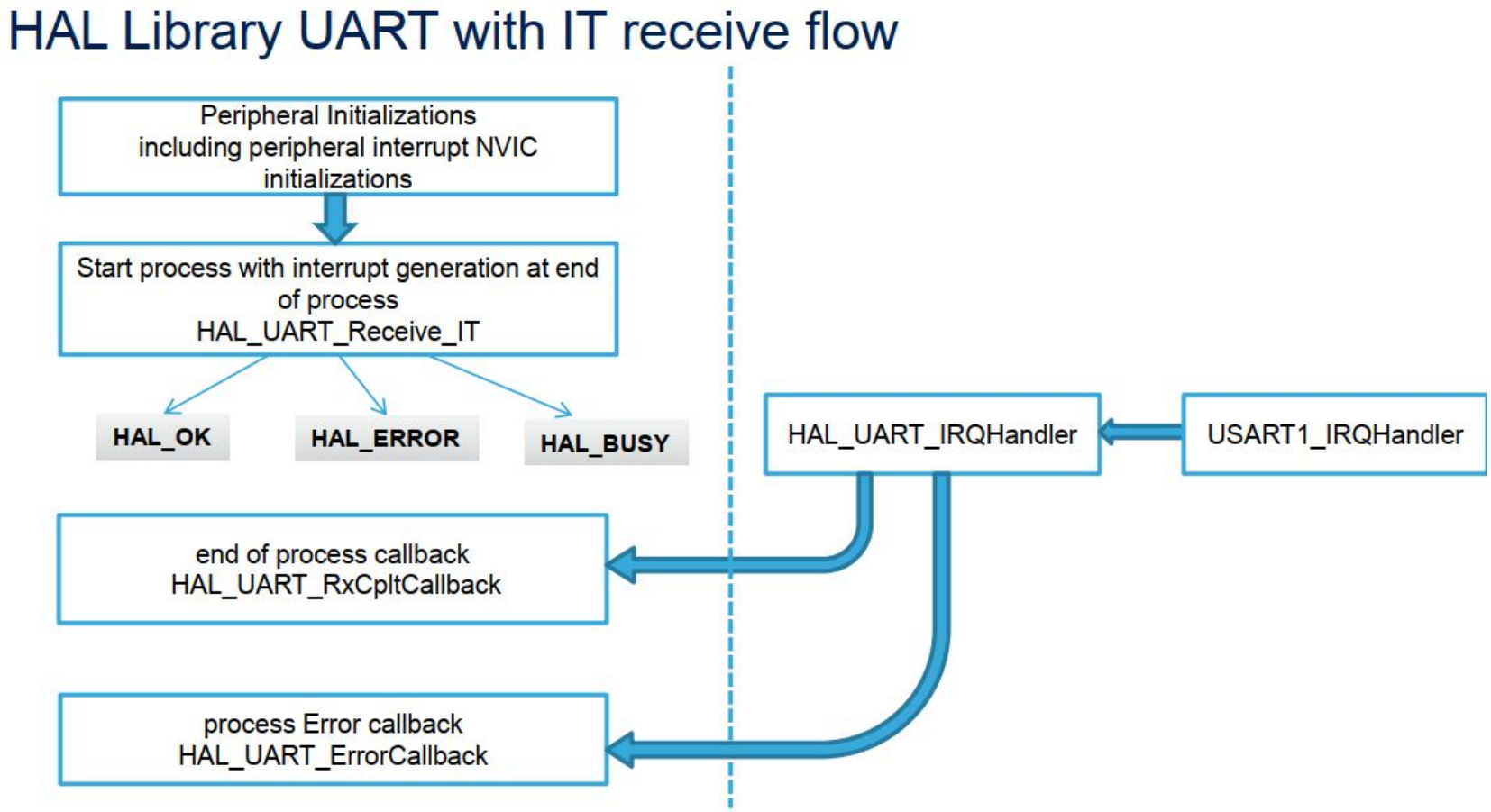


MODULE 5

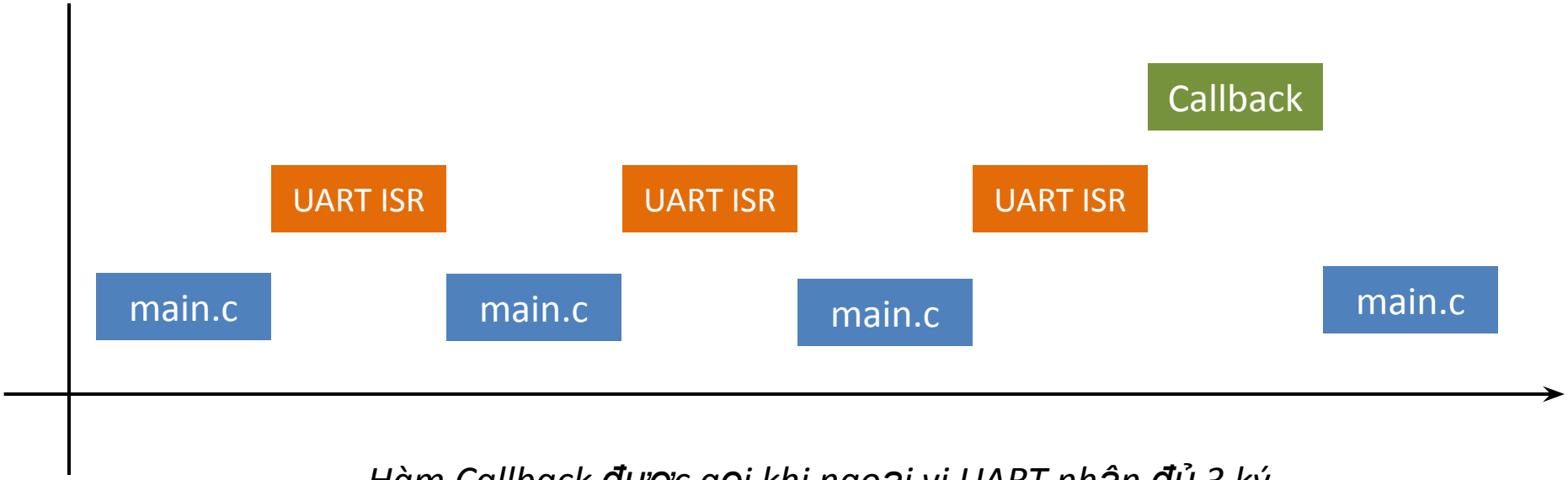
- THỰC HÀNH NHẬN DỮ LIỆU Ở CHẾ ĐỘ INTERRUPT



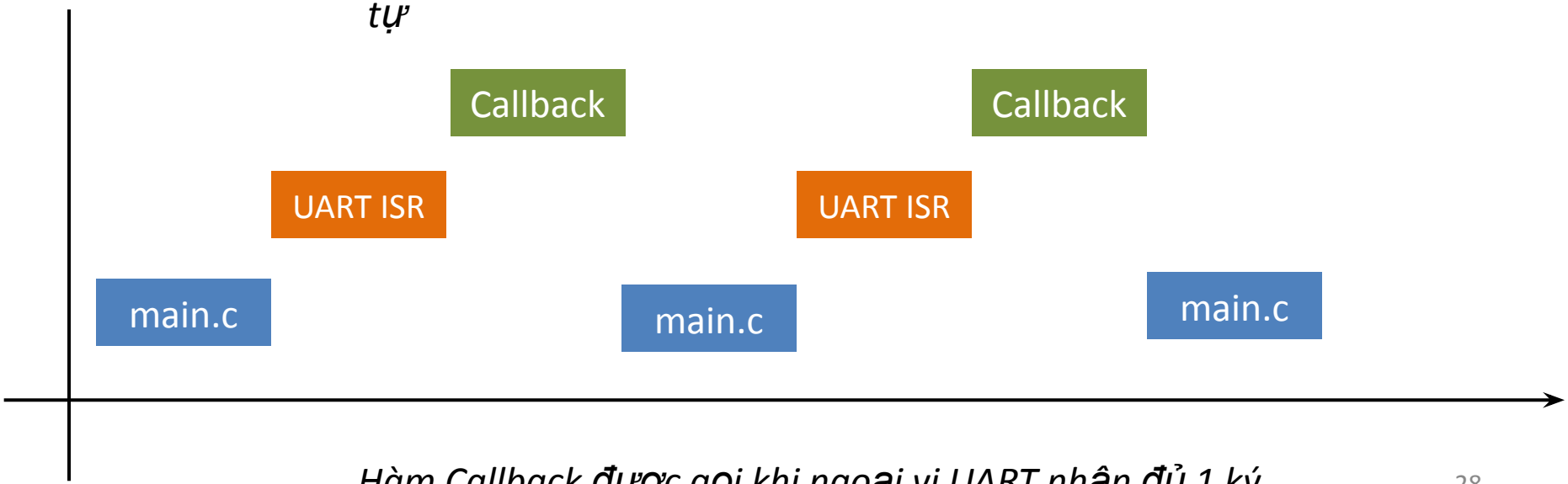
NHẬN DỮ LIỆU Ở CHẾ ĐỘ INTERRUPT



NHẬN DỮ LIỆU Ở CHẾ ĐỘ INTERRUPT

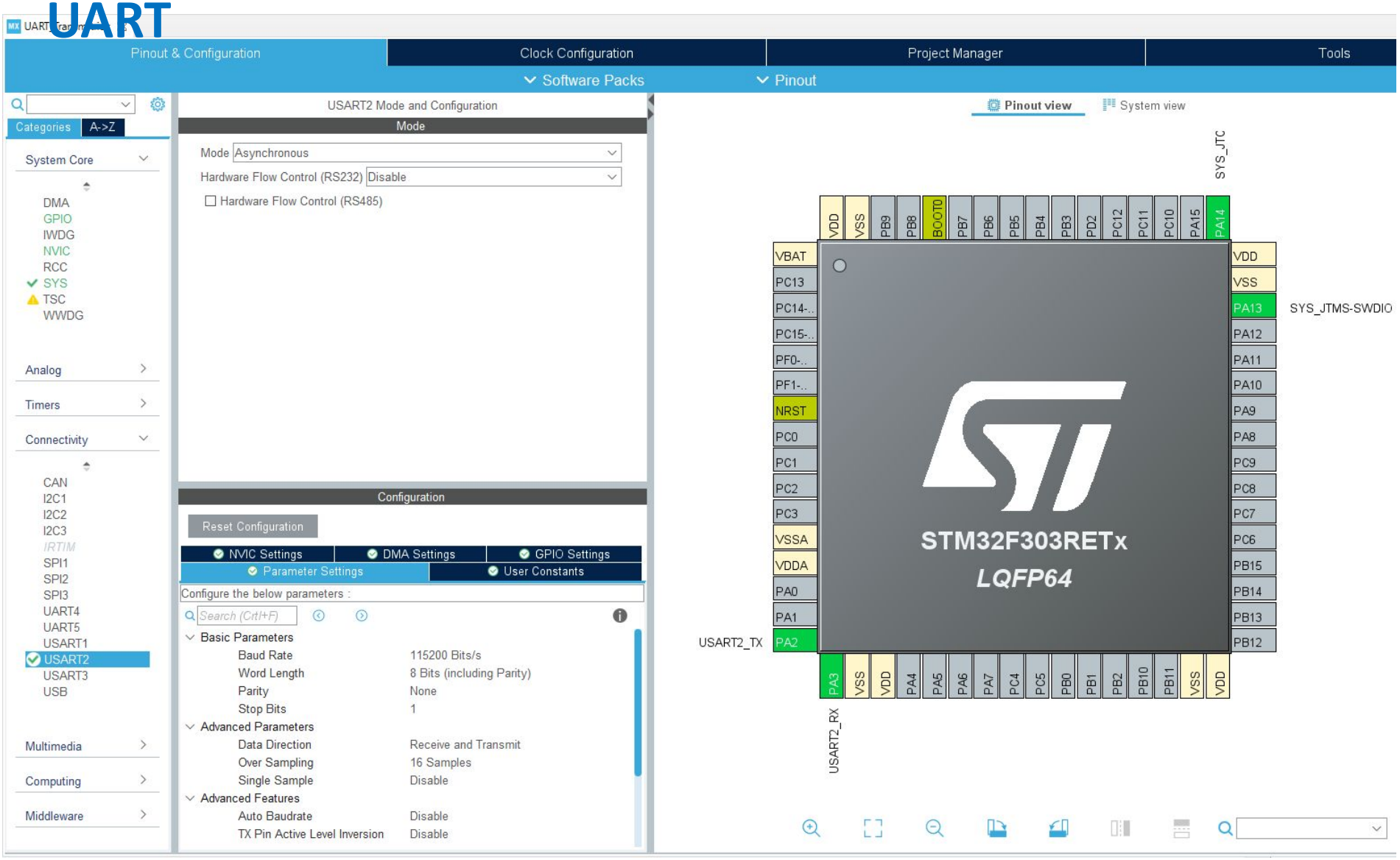


Hàm Callback được gọi khi ngoại vi UART nhận đủ 3 ký tự

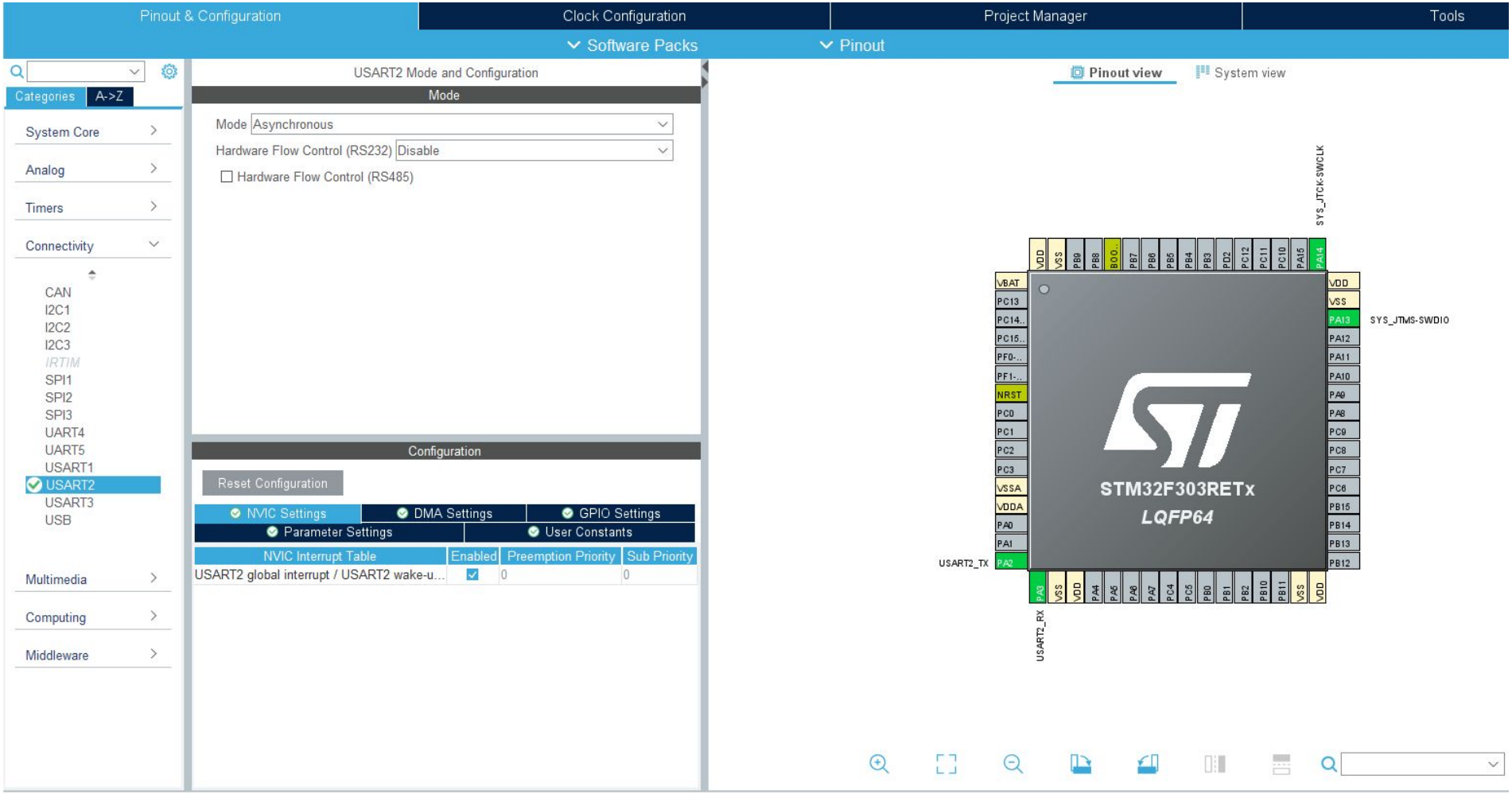


Hàm Callback được gọi khi ngoại vi UART nhận đủ 1 ký tự

CẤU HÌNH NGOẠI VI UART



CẤU HÌNH NGOẠI VI
UART



VIẾT CHƯƠNG TRÌNH & NẠP CODE

```
/* USER CODE BEGIN PV */
char Rx_byte = 0;
char Rx_buffer[10] = {0};
uint8_t Rx_index = 0;

/* USER CODE BEGIN 0 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    Rx_buffer[Rx_index] = Rx_byte;
    Rx_index ++;
    if(Rx_index >= 10)
    {
        //HAL_UART_Transmit(&huart2, Rx_buffer, 10, 1000);
        for(uint8_t i = 0; i < 10; i++)
         {
            Rx_buffer[i] = 0;  // Clear buffer
         }
        Rx_index = 0;
    }
    HAL_UART_Receive_IT(&huart2, (uint8_t*)&Rx_byte, 1);
}  /* USER CODE BEGIN 2 */
   HAL_UART_Receive_IT(&huart2, (uint8_t*)Rx_byte, 1);
```

LƯU Ý: KHÔNG COPY CODE - NGUY CƠ LỖI

Gửi chuỗi dữ liệu 01234567
Lần gửi đầu tiên: lưu đủ 01234567
Lần gửi thứ 2: 012345678
Lưu 01 => Đầy bộ đệm.
In dữ liệu lên màn hình và xóa bộ đệm
Trong lúc gửi byte đầu tiên lên màn hình
shift nhận thêm 1 byte dữ liệu có giá trị
lên thanh ghi RDR, bit RXNE được bật lê
Lúc này, vi xử lý điều khiển uart tiếp tục
màn hình, thì vi điều khiển không thể và
tiếp để đọc dữ liệu từ thanh ghi RDR, m
liệu từ RDR thì không thực hiện xóa bit
Mà trong lúc bit RXNE chưa được xóa m
tiếp tục nhận dữ liệu.
Nhưng không thể đẩy dữ liệu tới thanh
Dẫn đến các bye sau đó cũng không thể
thanh ghi RDR, thanh ghi RDR vẫn giữ gi

Cho đến khi, vi xử lý thực hiện xong việ
xong việc xóa bộ đệm, xong việc khai bá
từ ngắt UART, rồi thoát khỏi callback, th
vào lại callback, và lúc này sẽ nhận được
là '2'. Các byte khác sẽ mất, không nhận

UART Overrun error

The RXNE bit is set to indicate that the content of the shift register is transferred to the RDR
An interrupt is generated if the RXNEIE bit is set.

Clearing the RXNE bit is performed by a software read to the USART_RDR register

The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Data can not be transferred from the **shift register** to the R**DR register** until the RXNE bit is cleared.

https://www.st.com/resource/en/reference_manual/dm00043574-stm32f303xb-c-d-e-stm32f303x6-8-stm32f328x8-stm32f358xc-stm32f398xe-advanced-arm-based-mcus-stmicroelectronics.pdf

Viết chương trình để xử lý chuỗi nhận được qua UART từ Hercules Tool:

Nếu nhận được chuỗi “LEDON\r” thì cho LED sáng
Nếu nhận được chuỗi “LEDOFF\r” thì cho LED tắt
Nếu nhận được chuỗi “LEDBLINK\r” thì cho LED nhấp nháy 200ms

Sử dụng hàm xử lý chuỗi thư viện string.h
Tài liệu tham khảo:
https://tapit.vn/huong-dan-xu-ly-chuoi-trong-lap-trinh-vi-dieu-khien/
https://tapit.vn/huong-dan-xu-ly-chuoi-trong-lap-trinh-vi-dieu-khien-p2/

MỘT SỐ KINH NGHIỆM XỬ LÝ KHI GẶP PHẢI VẤN ĐỀ VỚI UART RECEIVE INTERRUPT

- Đặt breakpoint trong hàm Callback để kiểm tra vi xử lý có nhảy vào và thực thi trình phục vụ ngắt hay không mỗi khi có 1 ký tự, 1 byte dữ liệu đến.
- Thêm biến chứa dữ liệu và bộ đệm dữ liệu vào cửa sổ Live Expression để quan sát.
- Kiểm tra lại mã chương trình vừa viết nếu có vào ngắt nhưng thực hiện không như mong muốn.
- Nếu chương trình không dừng lại tại breakpoint thì các bạn cần:
  - Kiểm tra phần cứng trên cả 2 thiết bị.
  - Kiểm tra cấu hình khung truyền và tốc độ baud giữa 2 thiết bị có giống nhau không.
  - Kiểm tra đã cấu hình cho phép ngắt ở khối NVIC trong giao diện cấu hình CubeMX hay chưa
  - Kiểm tra đã gọi hàm HAL_UART_Receive_IT để khai báo nhận nhận dữ liệu trong hàm main hay chưa.
  - Kiểm tra các thanh ghi có liên quan như: cấu hình cho phép truyền hoặc nhận dữ liệu trong thanh ghi Status register (USART_SR), Data register (USART_DR), các cờ báo lỗi trong thanh ghi Control register 1 (USART_CR1)

MỘT SỐ LƯU Ý KHI VIẾT CHƯƠNG TRÌNH PHỤC VỤ NGẮT

- Viết chương trình xử lý ngắt càng ngắn gọn càng tốt. ASAP

  -> Nếu dài quá mà không gấp thì cho 1 biến nào đó lên 1, rồi trong main - while 1 kiểm tra biến đó rồi thực hiện chương trình

  -> Trong bài toán truyền nhận dữ liệu "batden100" "batden10". Cần thêm yếu tố là toàn vẹn dữ liệu : crc, checksum, kiểm tra dữ liệu khi nhận được, nếu toàn vẹn mới xử lý, nếu không, yêu cầu truyền lại hoặc bỏ gói tin.
- Cẩn thận khi sử dụng hàm HAL_Delay ở chương trình phục vụ ngắt.
- Khi làm việc với mảng thì nên quan tâm đến kích thước của mảng (phải chứa được dữ liệu dài nhất nhận được, và luôn luôn kiểm tra chỉ số so với chỉ số tối đa), xóa dữ liệu trong mảng mỗi khi tái sử dụng mảng đó.
- Nên tìm hiểu một số hàm làm việc với chuỗi trong thư viện **string.h**

Ngắt UART IDLE DMA

- Truyền 1 lượng lớn dữ liệu:
- Truyền file từ máy tính xuống
- Tải 1 file từ internet về (âm thanh, firmware, file cấu hình)
-

https://tapit.vn/huong-dan-su-dung-chuc-nang-uart-idle-dma/

DMA :
- Một master trên hệ thống bus

Giao tiếp với 1 thiết bị B.
Thiết bị B định kì 2s gửi ra 1 chuỗi dữ liệu không cố định kích cỡ, và cũng không có kí tự kết thúc.

Làm sao để biết là A đã nhận được 1 chuỗi hoàn chỉnh từ B?

- **Không dùng ngắt IDLE.**
- Không có kí tự kết thúc
- Không cố định?
- Dùng thêm 1 cái timer.
- Mỗi lần vào ngắt uart thì reset 1 biến thời gian.
- Và timer định kì kiểm tra biến thời gian.
- Nếu sau 1 khoản thời gian hơn thời gian nhận 1 byte mà biến k bị reset thì chứng tỏ không nhận thêm dữ liệu, chứng tỏ đã nhận 1 chuỗi hoàn chỉnh

Microphone
Thu âm thanh:
Tần số lấy mẫu âm thanh: 16KHz. -> 16.000 mẫu / 1s. 16ksps

Hệ thống yêu cầu khả năng realtime, vừa lấy âm thanh vừa xử lý DSP.

Nếu dùng vi xử lý để lấy mẫu thì vi xử lý không thể đi xử lý âm thanh được.

Dẫn đến cần dùng DMA để lấy mẫu âm thanh từ sensor micro và chuyển vào bộ nhớ.
VI xử lý sẽ phối hợp khi đủ lượng mẫu là xử lý ngay. (Ví dụ 16.000 mẫu xử lý 1 lần)

Dùng DMA. Khai báo 1 kích cỡ dữ liệu là 32.000 mẫu.
Và dùng ngắt Half transfer và ngắt full transfer.: