**I – Các hàm và kiểu dữ liệu liên quan tới buffer:**

**Buffer\_t:**

typedef struct Buffer\_type {

uint32\_t size; //

uint32\_t write\_idx; //position of write

uint32\_t read\_idx; //position of read, buffer is read will be free

uint8\_t\* Data; //buffer to save data, read from USART

uint8\_t Flags;

uint8\_t End\_String; //Character end of string

} Buffer\_t;

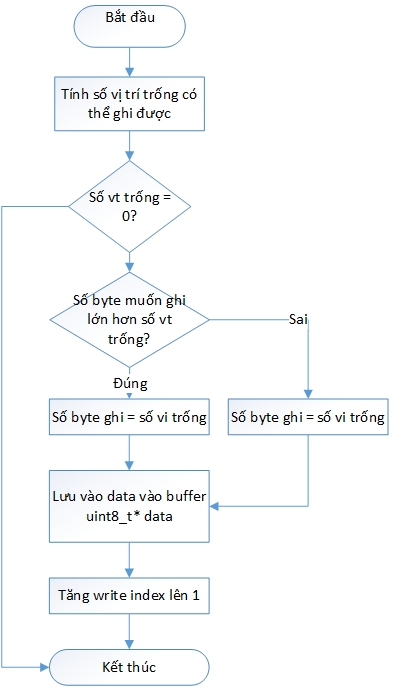
static Buffer\_t USART\_buffer;

static uint8\_t USART\_data[USART\_ESP8266\_SIZE];

**uint32\_t Buffer\_Write\_Free(Buffer\_t\* buff)**

**Chức năng:** Tính xem trong buffer còn bao nhiêu vị trí trống có thể ghi vào được. Nếu write\_idx > read\_idx => số vị trí trống = size – (write\_idx – read\_idx) -1. Nếu read\_idx > write\_idx => số vị trí trống = read\_idx – write\_idx – 1. Nếu write\_idx == read\_idx => tất cả đều đã được ghi, số vị trí trống = size.

**uint32\_t Buffer\_Write(Buffer\_t\* buff, uint32\_t count, uint8\_t\* dat):** ghi dữ liệu với vị trí bắt đầu được trở bởi \*dat, số byte là count tới buffer save data \*Data trong buff.



**void Buffer\_Reset(Buffer\_t\* buffer):** Reset write index và index về 0. (không reset data).

**uint32\_t Buffer\_Read\_String(Buffer\_t\* buffer, char \*save\_buff, uint32\_t count)** => đọc data nhận được trong USART.

* **Trường hợp không đọc trong buffer** khi số vị trí có thể đọc = 0 hoặc khi số vị trí đọc > 0 kết hợp với số vị trí có thể đọc < số byte muốn đọc và không tìm thấy kí tự kết thúc (ở đây là \n), chứng tỏ dữ liệu vẫn còn nằm trên đường truyền và chưa đọc lưu trong buffer.

**int32\_t Buffer\_Find\_Element(Buffer\_t\* buffer, uint8\_t Element):** tìm trong data của buffer có kí tự element hay không. Nếu có thì trả về vị trí của nó trong data của buffer, nếu không thì trả về -1. Ví dụ:

data của buffer là: “I will become a milionaire in five next year.” Character cần tìm là b => giá trị trả về của hàm là 7.

**II- Structure and data type của ESP8266:**

typedef struct {

uint8\_t IP[4];

uint8\_t MAC[4];

} ESP8266\_Connected\_Station\_t;

typedef struct {

ESP8266\_Connected\_Station\_t Stations[ESP8266\_MAX\_CONNECTION];

uint8\_t count;

} ESP8266\_Connected\_Multi\_Station\_t;

ESP8266\_Connected\_Station\_t**:** struct lưu địa chỉ IP và MAC của 1 station khi ESP8266 là access point.

ESP8266\_Connected\_Multi\_Station\_t **:** struct lưu địa chỉ IP và MAC của nhiều station, số station lớn nhất kết nối vào là **ESP8266\_MAX\_CONNECTION**.

typedef enum {

ESP8266\_ECN\_OPEN = 0x0,

ESP8266\_ECN\_WEP,

ESP8266\_ECN\_WPA\_PSK,

ESP8266\_ECN\_WPA2\_PSK,

ESP8266\_ECN\_WPA\_WPA2\_PSK

} Encrypt\_Method\_t;

Encrypt\_Method\_t: kiểu enum chỉ các kiểu bảo mật Wifi

1. **uint8\_t Hex\_To\_Num(char ch)**:

**Chức năng**: chuyển từ kí tự kiểu hexa nhận được (ví dụ ‘0’ – ‘9’, ‘a’ – ‘f’ hoặc ‘A’ – ‘F’) sang kí tự số (‘0’ → 0, ‘1’ → 1, … , ‘A’→ 10, ‘B’ →11). Vì khi nhận được dữ liệu trên USART thì chỉ nhận được kí tự số, không phải số.

**Input**: kí tự kiểu char

**Output:** số nguyên không dấu 8 bit

1. **uint8\_t Char\_Is\_Hex(char ch):**

**Chức năng**: Xét xem kí tự số có phải thuộc hexa hay không (nằm trong các kí tự 0 - 9, a - f hoặc A - F).

**Input**: kí tự kiểu char

**Output:** 1 nếu thuộc kiểu hexa và 0 nếu không thuộc kiểu hexa.

1. **uint32\_t Cal\_Hex\_Num(char\* ptr, uint8\_t\* count):**

**Chức năng**: Trả về kết quả là một số nguyên không dấu 32 bit từ một chuỗi (ptr) đưa vào, đồng thời biến count sẽ lưu giữ số kí số của ptr.

**Input**: char\* ptr - con trỏ chỉ tới chuỗi cần xử lý

uint8\_t\* count: con trỏ kiểu số nguyên không dấu 8 bit lưu giữ số kí số

**Output:** Kết quả theo dạng thập phân của chuỗi ptr.

Ví dụ: với ptr là con trỏ chỉ tới chuỗi “12345” => giá trị trả về từ hàm là 0x12345, count sẽ có giá trị là 5

1. **void ParseIP(char\* ip\_str, uint8\_t\* arr, uint8\_t\* cnt):**

**Chức năng:** Phân tách địa chỉ IP dưới dạng chuỗi nhận được từ USART và lưu vào mảng arr. Variable cnt dùng để tính số kí tự IP nhận được, biến này được dùng trong phân tích 1 chuỗi ký tự nhận được có chứa IP để đưa con trỏ tới vị trí cần xử lý tiếp theo sau chuỗi IP. Ví dụ với lệnh AT+CWJAP\_CUR? Response là +CWJAP\_CUR:<ssid>,<bssid>,<channel>,<rssi>. Giả sử ssid = “abc”, <bssid> = “ca:d7:19:d8:a6:44”, <channel> = 2, <rssi> = 4, ta có response là

+CWJAP\_CUR:”abc”,”ca:d7:19:d8:a6:44”,2,4. Đặt ptr là con trỏ tại vị trí a trong SSID, khi phân tích lấy được ssid => con trỏ sẽ được cộng lên 2, xử lý tiếp để bỏ qua chuỗi “,” tới vị trí MAC đầu tiên (ca), lúc này cnt sẽ lưu số ký tự của MAC là

#define ESP8266\_RETURN\_STATUS(ESP8266, status) \

ESP8266->last\_result = status; \

return status;

ESP8266: return status if eveything is OK. (last code line is ESP8266\_RETURN\_STATUS(ESP8266, ESP8266\_OK);