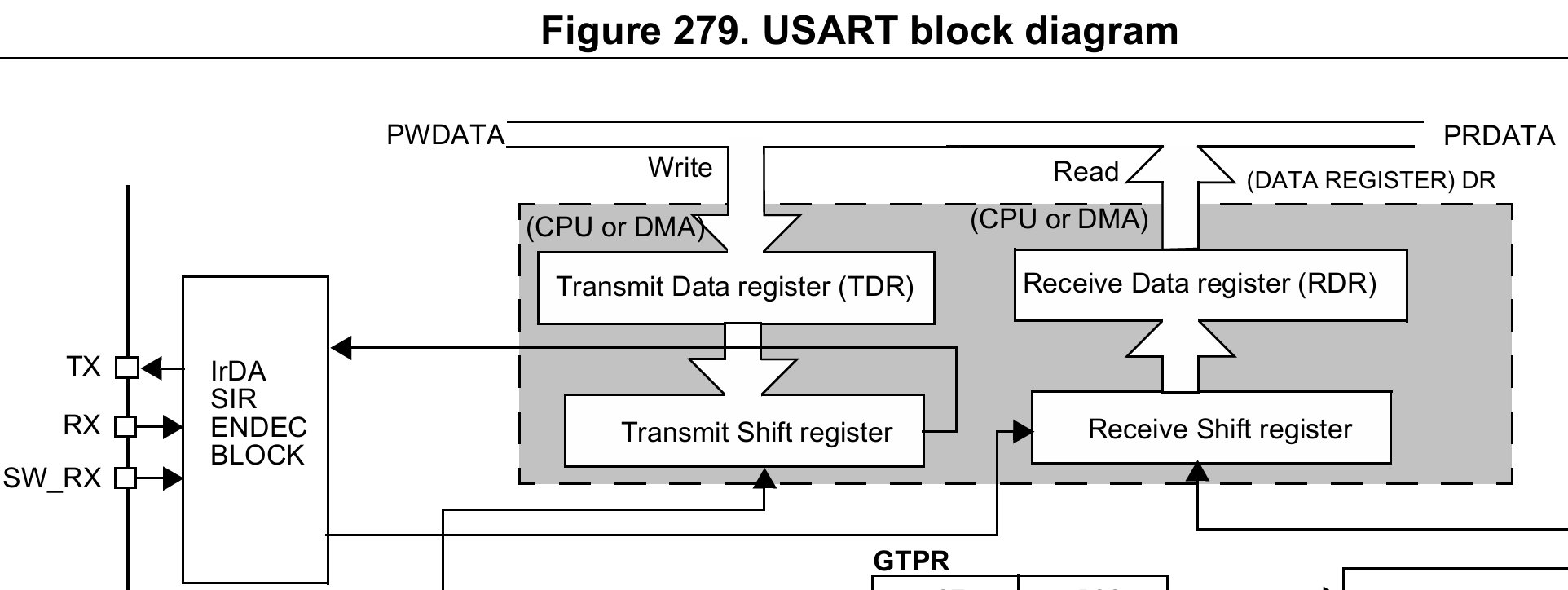
|  |  |
| --- | --- |
| TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TPHCM  **KHOA CƠ KHÍ CHẾ TẠO MÁY**  **BỘ MÔN CƠ ĐIỆN TỬ**  **Họ và tên: Võ Hồng Quân**  **Mã số SV: 22134012** | **BÁO CÁO**  **THỰC TẬP LẬP TRÌNH VI XỬ LÝ**  Ngày:  Tuần học: **06**  Nội dung: **Truyền nhận dữ liệu qua UART** |

1. **Khảo sát mô-đun truyền nhận dữ liệu UART:**

****

Universal Asynchronous Receiver/Transmitter is the hardware circuitry (module) being used for serial communication. UART is sold/shipped as a standalone integrated circuit (IC) or as an internal module within microcontrollers.

**UART** – Universal Asynchronous Receiver/Transmitter

**USART** – Universal Synchronous/Asynchronous Receiver/Transmitter

* 1. **HAL\_UART\_Transmit**(UART\_HandleTypeDef \* huart, const uint8\_t \* pData, uint16\_t Size, uint32\_t Timeout);

Giải thích lệnh:

Hàm HAL\_UART\_Transmit() truyền dữ liệu UART theo blocking, chờ hoàn tất trước khi tiếp tục. Nó gửi Size byte từ buffer pData qua UART huart, với thời gian chờ tối đa Timeout (ms). Nếu quá thời gian, hàm sẽ báo lỗi timeout.Hàm sẽ truyền từng byte trong buffer pData qua cổng UART theo chế độ blocking, chương trình sẽ dừng lại tại hàm này cho đến khi toàn bộ dữ liệu được truyền hoặc đã vượt quá thời gian timeout.

* 1. **HAL\_UART\_Receive**(UART\_HandleTypeDef \* **huart**, uint8\_t \***pData**, uint16\_t **Size**, uint32\_t **Timeout**);

Giải thích lệnh:

Hàm HAL\_UART\_Receive() nhận dữ liệu UART theo chế độ blocking, chờ đến khi nhận đủ byte hoặc timeout. Nó lưu dữ liệu vào buffer pData, với số byte xác định bởi Size, và dừng nếu hết thời gian Timeout. Trả về HAL\_OK nếu nhận thành công, HAL\_TIMEOUT nếu hết thời gian chờ, hoặc HAL\_ERROR nếu có lỗi

* 1. **HAL\_UART\_Transmit\_IT**(UART\_HandleTypeDef \* huart, const uint8\_t \* pData, uint16\_t Size);

 Hàm HAL\_UART\_Transmit\_IT() truyền dữ liệu UART theo chế độ non-blocking (ngắt), cho phép CPU thực hiện tác vụ khác trong khi truyền. Nó gửi dữ liệu từ buffer pData với kích thước Size, sau đó kích hoạt ngắt UART để xử lý truyền. Khi hoàn thành, hàm callback HAL\_UART\_TxCpltCallback() sẽ được gọi. Ví dụ: HAL\_UART\_Transmit\_IT(&huart1, msg, sizeof(msg)-1); sẽ truyền dữ liệu trong nền và tiếp tục chương trình mà không chờ hoàn tất.

* 1. **HAL\_UART\_Receive\_IT**(UART\_HandleTypeDef \* huart, uint8\_t \* pData, uint16\_t Size);

Hàm HAL\_UART\_Receive\_IT() nhận dữ liệu UART theo chế độ interrupt (ngắt), không chặn chương trình. Nó nhận Size byte và lưu vào pData, sau đó tiếp tục chương trình chính. Khi nhận xong, hàm callback HAL\_UART\_RxCpltCallback() sẽ được gọi để xử lý dữ liệu.

void **HAL\_UART\_RxCpltCallback**(UART\_HandleTypeDef \*huart) {

// Handle UART RX Interrupt Here! }

Giải thích lệnh:

Hàm HAL\_UART\_RxCpltCallback() là callback function được gọi tự động khi UART nhận xong dữ liệu trong chế độ ngắt (HAL\_UART\_Receive\_IT()). Dùng bằng cách ghi đè (override) hàm này để xử lý dữ liệu nhận được, như lưu vào buffer hoặc phản hồi ngay sau khi nhận xong.

void **HAL\_UART\_TxCpltCallback**(UART\_HandleTypeDef \*huart) { // Handle UART TX Interrupt Here! }

Giải thích lệnh:

Hàm HAL\_UART\_TxCpltCallback() là một callback function được gọi tự động khi quá trình truyền dữ liệu UART (theo ngắt) hoàn tất. Hàm này giúp thực hiện các tác vụ sau khi dữ liệu đã được gửi hết mà không làm gián đoạn chương trình chính. Người dùng có thể ghi đè (override) hàm này để xử lý các hành động như gửi tiếp dữ liệu hoặc báo hiệu quá trình truyền đã xong

* 1. **Retargeting printf và scanf:**

|  |
| --- |
| struct \_\_FILE {  int handle;  };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1,(uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  FILE \_\_stdin;  int fgetc(FILE \*f) {  uint8\_t ch = 0;  \_\_HAL\_UART\_CLEAR\_OREFLAG(&huart1);  HAL\_UART\_Receive(&huart1,(uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  } |

* Nêu cú pháp sử dụng và giải thích lệnh printf:

Cú pháp: printf("Chuỗi định dạng", tham\_số1, tham\_số2, ...);

Giải thích: Hàm printf in dữ liệu ra màn hình hoặc thiết bị đầu ra, nhưng trên STM32, không có màn hình console, nên ta phải retarget (định tuyến lại) để gửi dữ liệu qua UART bằng cách định nghĩa lại fputc().

* Nêu cú pháp sử dụng và giải thích lệnh scanf

Cú pháp: scanf("Chuỗi định dạng", &biến1, &biến2, ...);

Giải thích: Hàm scanf đọc dữ liệu từ đầu vào (mặc định là bàn phím), nhưng trên STM32, ta phải retarget để đọc dữ liệu từ UART bằng cách định nghĩa lại fgetc().

1. **Lập trình ứng dụng**
   1. **Sử dụng printf và scanf kết hợp với UART**

Lập trình dự án STM32 sử dụng UART và thư viện printf, scanf để:

* In một biến Count từ 0 đến 10 rồi lặp lại bằng lệnh printf.
* Nhận một biến bằng lệnh scanf rồi in lại biến đã nhận bằng printf
* Nhận một chuỗi bằng lệnh scanf rồi in lại chuỗi đã nhận bằng printf

Bài làm:

|  |
| --- |
| #include "main.h"  #include <stdio.h>  #include <string.h>  UART\_HandleTypeDef huart1;  /\* Retarget printf and scanf \*/  struct \_\_FILE {  int handle;  };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  FILE \_\_stdin;  int fgetc(FILE \*f) {  uint8\_t ch = 0;  \_\_HAL\_UART\_CLEAR\_OREFLAG(&huart1);  HAL\_UART\_Receive(&huart1, &ch, 1, HAL\_MAX\_DELAY);  return ch;  }  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_USART1\_UART\_Init(void);  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  int count = 0;  int num;  char str[20];  while (1) {  // In biến Count từ 0 đến 10  printf("Count: %d\n", count);  count = (count + 1) % 11; // Reset về 0 khi đạt 10  HAL\_Delay(1000); // Delay 1s  // Nhận số từ UART và in lại  printf("Nhập một số: ");  scanf("%d", &num);  printf("Số đã nhập: %d\n", num);  // Nhận chuỗi từ UART và in lại  printf("Nhập một chuỗi: ");  scanf("%s", str);  printf("Chuỗi đã nhập: %s\n", str);  }  }  /\* UART Initialization \*/  static void MX\_USART1\_UART\_Init(void) {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  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();  }  } |

* 1. **Gởi sự kiến nhấn nút qua UART:**

**Lập trình dự án STM32 để gởi sự kiện nhấn nút Px lên máy tính như sau:**

* Ngay khi người sử dụng nhấn nút P1 thì vi điều khiển gởi lên máy tính "P1 on\n"
* Ngay khi người sử dụng nhấn nút P2 thì vi điều khiển gởi lên máy tính "P2 on\n"
* Ngay khi người sử dụng nhấn nút P3 thì vi điều khiển gởi lên máy tính "P3 on\n".
* Ngay khi người sử dụng nhấn nút P4 thì vi điều khiển gởi lên máy tính "P4 on\n".
* Ngay khi người sử dụng thả tay ra sau khi nhấn nút P1 thì vi điều khiển gởi lên máy tính "P1 off\n".
* Ngay khi người sử dụng thả tay ra sau khi nhấn nút P2 thì vi điều khiển gởi lên máy tính "P2 off\n".
* Ngay khi người sử dụng thả tay ra sau khi nhấn nút P3 thì vi điều khiển gởi lên máy tính "P3 off\n".
* Ngay khi người sử dụng thả tay ra sau khi nhấn nút P4 thì vi điều khiển gởi lên máy tính "P4 off\n"

Gợi ý: Sử dụng state machine để đọc các trạng thái nút nhấn. Kết hợp printf để gởi thông tin trạng thái nút nhấn.

Bài làm:

|  |
| --- |
| #include "main.h"  #include <stdio.h>  UART\_HandleTypeDef huart1;  #define BUTTON\_PRESSED GPIO\_PIN\_RESET // Trạng thái nút nhấn (nút nhấn kéo xuống GND)  #define BUTTON\_RELEASED GPIO\_PIN\_SET // Trạng thái nút thả  // Biến lưu trạng thái trước đó của các nút  uint8\_t prev\_P1 = BUTTON\_RELEASED;  uint8\_t prev\_P2 = BUTTON\_RELEASED;  uint8\_t prev\_P3 = BUTTON\_RELEASED;  uint8\_t prev\_P4 = BUTTON\_RELEASED;  /\* Retarget printf to UART \*/  struct \_\_FILE {  int handle;  };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  /\* Function prototypes \*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_USART1\_UART\_Init(void);  void CheckButtonState(GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin, uint8\_t \*prev\_state, const char \*button\_name);  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  while (1) {  CheckButtonState(P1\_GPIO\_Port, P1\_Pin, &prev\_P1, "P1");  CheckButtonState(P2\_GPIO\_Port, P2\_Pin, &prev\_P2, "P2");  CheckButtonState(P3\_GPIO\_Port, P3\_Pin, &prev\_P3, "P3");  CheckButtonState(P4\_GPIO\_Port, P4\_Pin, &prev\_P4, "P4");  }  }  /\* Check button state and send event via UART \*/  void CheckButtonState(GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin, uint8\_t \*prev\_state, const char \*button\_name) {  uint8\_t current\_state = HAL\_GPIO\_ReadPin(GPIOx, GPIO\_Pin);  if (current\_state == BUTTON\_PRESSED && \*prev\_state == BUTTON\_RELEASED) {  printf("%s on\n", button\_name);  }  if (current\_state == BUTTON\_RELEASED && \*prev\_state == BUTTON\_PRESSED) {  printf("%s off\n", button\_name);  }  \*prev\_state = current\_state; // Cập nhật trạng thái trước đó  }  /\* UART Initialization \*/  static void MX\_USART1\_UART\_Init(void) {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  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();  }  } |

* 1. **Gởi sự kiến nhấn nút qua UART:**

Lập trình dự án vi điều khiển đọc lệnh từ máy tính thông qua cổng UART như sau:

* Khi nhận được lệnh từ máy tính "**D1 on\n**" thì đèn **D1** sáng.
* Khi nhận được lệnh từ máy tính "**D2 on\n**" thì đèn **D2** sáng.
* Khi nhận được lệnh từ máy tính "**D1 off\n**" thì đèn **D1** tắt.
* Khi nhận được lệnh từ máy tính "**D2 off\n**" thì đèn **D2** tắt.

Bài làm:

|  |
| --- |
| #include "main.h"  #include <stdio.h>  #include <string.h>  UART\_HandleTypeDef huart1;  #define RX\_BUFFER\_SIZE 16  uint8\_t rx\_buffer[RX\_BUFFER\_SIZE]; // Bộ đệm nhận dữ liệu UART  uint8\_t rx\_index = 0; // Vị trí hiện tại trong buffer  /\* Retarget printf \*/  struct \_\_FILE {  int handle;  };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  /\* Function prototypes \*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_USART1\_UART\_Init(void);  void ProcessCommand(void);  /\* UART RX Interrupt Callback \*/  void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) {  if (huart->Instance == USART1) {  if (rx\_buffer[rx\_index] == '\n') { // Kiểm tra nếu kết thúc chuỗi lệnh  rx\_buffer[rx\_index + 1] = '\0'; // Đảm bảo chuỗi kết thúc bằng NULL  ProcessCommand(); // Xử lý lệnh  rx\_index = 0; // Reset vị trí nhận dữ liệu  } else {  rx\_index = (rx\_index + 1) % RX\_BUFFER\_SIZE;  }  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Tiếp tục nhận dữ liệu  }  }  /\* Process received UART command \*/  void ProcessCommand(void) {  if (strcmp((char \*)rx\_buffer, "D1 on\n") == 0) {  HAL\_GPIO\_WritePin(D1\_GPIO\_Port, D1\_Pin, GPIO\_PIN\_SET);  printf("D1 turned ON\n");  } else if (strcmp((char \*)rx\_buffer, "D1 off\n") == 0) {  HAL\_GPIO\_WritePin(D1\_GPIO\_Port, D1\_Pin, GPIO\_PIN\_RESET);  printf("D1 turned OFF\n");  } else if (strcmp((char \*)rx\_buffer, "D2 on\n") == 0) {  HAL\_GPIO\_WritePin(D2\_GPIO\_Port, D2\_Pin, GPIO\_PIN\_SET);  printf("D2 turned ON\n");  } else if (strcmp((char \*)rx\_buffer, "D2 off\n") == 0) {  HAL\_GPIO\_WritePin(D2\_GPIO\_Port, D2\_Pin, GPIO\_PIN\_RESET);  printf("D2 turned OFF\n");  } else {  printf("Unknown command: %s", rx\_buffer);  }  }  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  printf("UART Ready. Send 'D1 on', 'D1 off', 'D2 on', 'D2 off' to control LEDs.\n");  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Bắt đầu nhận dữ liệu qua UART  while (1) {  // Chương trình chính không cần xử lý gì vì UART xử lý qua interrupt  }  }  /\* UART Initialization \*/  static void MX\_USART1\_UART\_Init(void) {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  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();  }  } |

* 1. **Lập trình truyền nhận UART theo cơ chế ngắt kết hợp điều khiển LED 7 đoạn:**

Lập trình hệ thống vi điều khiển STM32 nhận lệnh từ máy tính thông qua UART:

* "**D12345\n**" : Cập nhật hiển thị LED là 12345.

(Cú pháp: <D><LED\_val><\n>)

* "**D123\n**": Cập nhật hiển thị LED là 123. (tắt 2 LED đầu tiên)
* "**D0123\n**": Cập nhật hiển thị LED là 0123. (tắt hiển thị 1 LED đầu tiên, LED số 2 hiển thị 0)

Gợi ý: Sử dụng ngắt timer để quét hiển thị LED 7 đoạn ở bài trước.

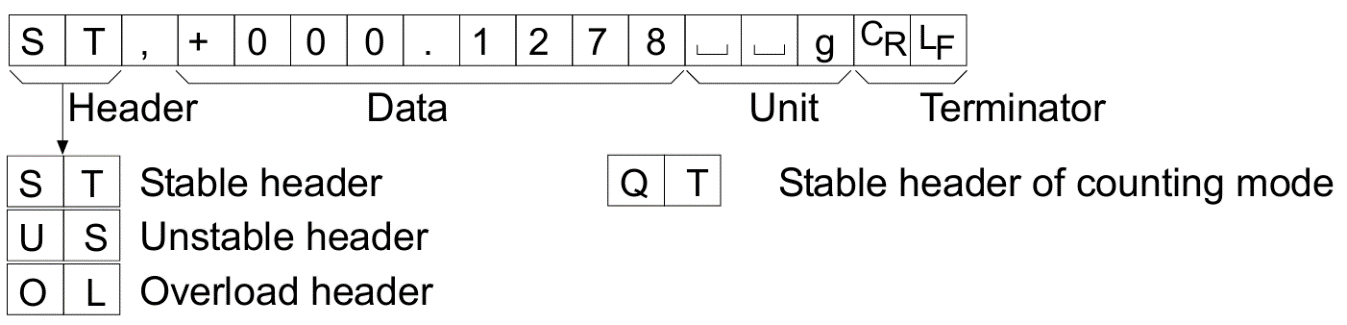
Bài làm:

|  |
| --- |
| #include "main.h"  #include <stdio.h>  #include <string.h>  UART\_HandleTypeDef huart1;  TIM\_HandleTypeDef htim1;  #define RX\_BUFFER\_SIZE 10  uint8\_t rx\_buffer[RX\_BUFFER\_SIZE]; // Buffer nhận UART  uint8\_t rx\_index = 0; // Chỉ số buffer nhận  uint8\_t led\_display[5] = "00000"; // Chuỗi hiển thị LED 7 đoạn  /\* Retarget printf \*/  struct \_\_FILE { int handle; };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  /\* Function prototypes \*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_USART1\_UART\_Init(void);  static void MX\_TIM1\_Init(void);  void ProcessCommand(void);  void LED\_Update(void);  /\* UART RX Interrupt Callback \*/  void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) {  if (huart->Instance == USART1) {  if (rx\_buffer[rx\_index] == '\n') { // Nếu nhận được '\n' thì xử lý lệnh  rx\_buffer[rx\_index + 1] = '\0'; // Kết thúc chuỗi  ProcessCommand(); // Xử lý lệnh  rx\_index = 0; // Reset vị trí buffer  } else {  rx\_index = (rx\_index + 1) % RX\_BUFFER\_SIZE;  }  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Tiếp tục nhận dữ liệu  }  }  /\* Xử lý lệnh nhận được \*/  void ProcessCommand(void) {  if (rx\_buffer[0] == 'D') { // Kiểm tra lệnh bắt đầu bằng 'D'  uint8\_t len = strlen((char \*)rx\_buffer) - 2; // Số ký tự số (bỏ 'D' và '\n')    // Xóa buffer hiển thị  memset(led\_display, ' ', 5);  // Ghi số vào cuối buffer để canh phải  for (int i = 0; i < len; i++) {  led\_display[5 - len + i] = rx\_buffer[i + 1]; // Chép số vào buffer  }  printf("LED updated: %s\n", led\_display);  }  }  /\* Ngắt Timer quét LED 7 đoạn \*/  void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) {  if (htim->Instance == TIM1) {  LED\_Update();  }  }  /\* Cập nhật LED 7 đoạn \*/  void LED\_Update(void) {  uint8\_t pos[5] = {0x08, 0x10, 0x20, 0x40, 0x80}; // Mặt nạ LED  static uint8\_t current\_digit = 0;  uint8\_t code = (led\_display[current\_digit] == ' ') ? LED\_OFF : LED\_decode\_char(led\_display[current\_digit]);    uint16\_t d16 = ((uint16\_t)pos[current\_digit] << 8) | code;  Data16\_put(d16);  current\_digit = (current\_digit + 1) % 5; // Chuyển qua LED kế tiếp  }  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  MX\_TIM1\_Init();  printf("UART Ready. Send 'D12345', 'D123' or 'D0123' to update LED display.\n");  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Bắt đầu nhận UART  HAL\_TIM\_Base\_Start\_IT(&htim1); // Bắt đầu Timer ngắt quét LED  while (1) {  // Chương trình chính để trống, mọi thứ xử lý qua ngắt  }  }  /\* UART Initialization \*/  static void MX\_USART1\_UART\_Init(void) {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  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();  }  }  /\* Timer Initialization \*/  static void MX\_TIM1\_Init(void) {  htim1.Instance = TIM1;  htim1.Init.Prescaler = 999;  htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  htim1.Init.Period = 15999;  htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  htim1.Init.RepetitionCounter = 0;  htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;  if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK) {  Error\_Handler();  }  HAL\_TIM\_Base\_Start\_IT(&htim1); // Bật ngắt Timer  } |

* 1. **Lập trình truyền nhận UART giao tiếp đầu cân:**

Một đầu cân tự động xuất tín hiệu qua cổng UART theo các qui ước sau đây:

**Gói dữ liệu**: Gồm 4 phần như mô tả dưới đây.



* Phần Header: 2 ký tự (ST, US, OL, QT)
* Phần Data 9 ký tự, là khối lượng cân.
* Phần Unit: 3 ký tự (g, mg , kg)
* Phần Terminator: 2 ký tự CR, LF.

**Lập trình hệ thống STM32 giao tiếp với cân và máy tính như sau:**

* Mỗi khi cân gởi gói dữ liệu đến vi điều khiển, thì vi điều khiển nhận gói dữ liệu và phân tích:
  + Nếu header là **Stable**, vi điều khiển sẽ tách lấy giá trị khối lượng rồi hiển thị dòng trên là Stable; dòng dưới là khối lượng.
  + Nếu header là **Unstable**, vi điều khiển sẽ tách lấy giá trị khối lượng rồi hiển thị dòng trên là Unstable; dòng dưới là khối lượng.
  + Nếu header là Overload, vi điều khiển chỉ hiển thị dòng trên “Overload”; dòng dưới không hiển thị gì.
  + Các chế độ khác bỏ qua.
* Ví dụ:

|  |  |
| --- | --- |
| Chuỗi nhận được | Hiển thị LCD |
|  |  |
|  |  |
|  | Overload |

Bài làm:

|  |
| --- |
| #include "main.h"  #include <stdio.h>  #include <string.h>  #include "myLCD.h"  UART\_HandleTypeDef huart1;  #define RX\_BUFFER\_SIZE 16 // Độ dài buffer nhận dữ liệu UART  uint8\_t rx\_buffer[RX\_BUFFER\_SIZE]; // Buffer nhận dữ liệu UART  uint8\_t rx\_index = 0; // Chỉ số buffer nhận  /\* Retarget printf \*/  struct \_\_FILE { int handle; };  FILE \_\_stdout;  int fputc(int ch, FILE \*f) {  HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, HAL\_MAX\_DELAY);  return ch;  }  /\* Function prototypes \*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_USART1\_UART\_Init(void);  void ProcessScaleData(void);  /\* UART RX Interrupt Callback \*/  void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) {  if (huart->Instance == USART1) { // Kiểm tra UART1  if (rx\_buffer[rx\_index] == '\n') { // Nếu nhận đủ dữ liệu (CR, LF)  rx\_buffer[rx\_index + 1] = '\0'; // Kết thúc chuỗi  ProcessScaleData(); // Gọi hàm xử lý dữ liệu  rx\_index = 0; // Reset buffer  } else {  rx\_index = (rx\_index + 1) % RX\_BUFFER\_SIZE;  }  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Tiếp tục nhận dữ liệu UART  }  }  /\* Xử lý dữ liệu từ đầu cân \*/  void ProcessScaleData(void) {  char header[3] = {0};  char weight[10] = {0};  char unit[4] = {0};  sscanf((char \*)rx\_buffer, "%2s%9s%3s", header, weight, unit); // Tách dữ liệu  if (strcmp(header, "ST") == 0) { // Stable  LCD\_Clear();  LCD\_GotoXY(0, 0);  LCD\_PutString((uint8\_t \*)"Stable");  LCD\_GotoXY(0, 1);  LCD\_PutString((uint8\_t \*)weight);  LCD\_PutString((uint8\_t \*)unit);  }  else if (strcmp(header, "US") == 0) { // Unstable  LCD\_Clear();  LCD\_GotoXY(0, 0);  LCD\_PutString((uint8\_t \*)"Unstable");  LCD\_GotoXY(0, 1);  LCD\_PutString((uint8\_t \*)weight);  LCD\_PutString((uint8\_t \*)unit);  }  else if (strcmp(header, "OL") == 0) { // Overload  LCD\_Clear();  LCD\_GotoXY(0, 0);  LCD\_PutString((uint8\_t \*)"Overload");  }  else {  printf("Invalid Data: %s\n", rx\_buffer); // Nếu không đúng format  }  }  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  LCD\_Init();  printf("UART Scale Interface Ready\n");  HAL\_UART\_Receive\_IT(&huart1, &rx\_buffer[rx\_index], 1); // Bắt đầu nhận dữ liệu UART  while (1) {  // Chương trình chính để trống, mọi xử lý thông qua ngắt UART  }  }  /\* UART Initialization \*/  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();  }  } |