# **Quy Tắc Cấu Trúc Code Trong AUTOSAR Classic**

### **1. Giới thiệu**

Trong việc phát triển phần mềm nhúng, đặc biệt trên nền tảng **AUTOSAR Classic**, việc tuân thủ các quy tắc lập trình tốt là yếu tố then chốt để đảm bảo tính bảo trì và mở rộng của mã nguồn. Hệ thống điều khiển động cơ là một ví dụ phổ biến trong các dự án nhúng, và việc áp dụng các quy tắc lập trình sẽ giúp đảm bảo mã nguồn an toàn, dễ bảo trì, và hạn chế lỗi.

Tài liệu này nhằm cung cấp một hướng dẫn chi tiết và toàn diện về:

* Cấu trúc file header và file source.
* Quy tắc khai báo và sử dụng biến.
* Cách sử dụng từ khóa static và extern.
* Quy tắc viết comment để đảm bảo mã nguồn rõ ràng, dễ hiểu.
* Ví dụ cụ thể về tổ chức mã nguồn.

### **2. Cấu trúc và quy tắc đặt tên file**

#### **2.1 Cấu trúc file header (.h)**

File **header** (.h) là file chứa các khai báo mà các file source khác có thể sử dụng. Nó đóng vai trò là một "giao diện" giữa các phần của chương trình.

* Mỗi file header phải có **include guards** để ngăn chặn việc file bị include nhiều lần, gây ra lỗi biên dịch.
* Tên file header phải rõ ràng, ngắn gọn và mô tả chính xác chức năng của nó.

##### **Cấu trúc chung của file header:**

1. **Include Guards**:
   * Sử dụng #ifndef, #define, và #endif để bảo vệ file khỏi việc include nhiều lần.

| #ifndef MOTOR\_CONTROL\_H #define MOTOR\_CONTROL\_H // Nội dung file header #endif // MOTOR\_CONTROL\_H |
| --- |

1. **Phần mô tả (documentation block)**:
   * Bao gồm tên file, mô tả chức năng, tên tác giả, ngày tạo, và phiên bản. Phần này giúp những lập trình viên khác hiểu rõ về file này mà không cần đọc chi tiết mã.

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file MotorControl.h  \* @brief Khai báo các hàm và cấu trúc liên quan đến điều khiển động cơ  \* @details File này cung cấp giao diện cho việc điều khiển động cơ, bao gồm  \* khởi tạo, đặt tốc độ, và dừng động cơ.  \* @version 1.0  \* @date 2024-09-11  \* @author HALA Academy  \* @website https://hala.edu.vn/  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| --- |

1. **Include thư viện chuẩn (nếu cần)**:
   * Chỉ include những thư viện cần thiết, để tránh làm tăng kích thước mã nguồn và gây lỗi không mong muốn.

| #include <stdint.h> // Sử dụng cho kiểu dữ liệu cố định |
| --- |

1. **Khai báo kiểu dữ liệu (typedef, enum, struct)**:
   * File header là nơi khai báo các kiểu dữ liệu như struct, enum, và typedef để các file source khác có thể sử dụng.

| typedef enum {  MOTOR\_OFF = 0,  MOTOR\_ON = 1 } MotorState\_t;  typedef struct {  uint8\_t motorId; // ID của động cơ  uint16\_t motorSpeed; // Tốc độ hiện tại của động cơ  MotorState\_t motorState; // Trạng thái hiện tại của động cơ (ON hoặc OFF) } Motor\_t; |
| --- |

1. **Khai báo các hàm và biến**:
   * Các hàm và biến toàn cục được khai báo trong file header nhưng không định nghĩa chi tiết.

| extern Motor\_t motorList[10]; // Biến toàn cục dùng ở nhiều file  void Motor\_Init(uint8\_t motorId); // Khai báo hàm khởi tạo int Motor\_SetSpeed(uint8\_t motorId, uint8\_t speed); // Khai báo hàm đặt tốc độ |
| --- |

#### **2.2 Cấu trúc file source (.c)**

File **source** (.c) chứa phần định nghĩa chi tiết của các hàm được khai báo trong file header.

1. **Include file header và các thư viện cần thiết**:
   * File source luôn phải include các file header liên quan để sử dụng các khai báo và kiểu dữ liệu.

| #include "MotorControl.h" #include <stdio.h> // Thư viện chuẩn để in ra console |
| --- |

1. **Phần mô tả (documentation block)**:
   * Phần mô tả của file source giúp ghi chú chức năng của file, bao gồm thông tin tác giả, ngày tạo, và các phiên bản.

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file MotorControl.c  \* @brief Định nghĩa các hàm điều khiển động cơ.  \* @details File này chứa phần định nghĩa của các hàm điều khiển động cơ, bao gồm  \* khởi tạo, đặt tốc độ, và dừng động cơ.  \* @version 1.0  \* @date 2024-09-11  \* @author HALA Academy  \* @website https://hala.edu.vn/  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| --- |

1. **Định nghĩa các hàm**:
   * Trong file source, các hàm được định nghĩa chi tiết. Phần logic của hàm phải được viết rõ ràng, với comment giải thích chi tiết.

| void Motor\_Init(uint8\_t motorId) {  motorList[motorId].motorId = motorId;  motorList[motorId].motorSpeed = 0U;  motorList[motorId].motorState = MOTOR\_OFF;  printf("Motor %d initialized.\n", motorId); } |
| --- |

#### **2.3 Quy tắc đặt tên file**

* **File header** phải có hậu tố .h, và tên file phải mô tả chính xác chức năng của nó.
* **File source** phải có hậu tố .c, và tên file phải đồng nhất với file header tương ứng.

Ví dụ:

* **File header**: MotorControl.h
* **File source**: MotorControl.c

Điều này giúp lập trình viên dễ dàng nhận biết chức năng của từng file và các file liên quan với nhau.

### **3. Quy tắc khai báo biến và cách sử dụng**

#### **3.1 Biến cục bộ (Local Variables)**

* Biến cục bộ được khai báo bên trong hàm và chỉ có thể sử dụng trong hàm đó. Đây là cách tốt nhất để tránh xung đột biến giữa các hàm.
* Tên biến cục bộ phải rõ ràng và mô tả chính xác chức năng của nó.

**Ví dụ**:

| void Motor\_Start(uint8\_t motorId) {  uint8\_t speed = 50; // Biến cục bộ, chỉ dùng trong hàm này  // logic điều khiển động cơ } |
| --- |

#### **3.2 Biến toàn cục (Global Variables)**

* Biến toàn cục có thể truy cập từ bất kỳ nơi nào trong chương trình, nhưng việc sử dụng chúng nên được hạn chế.
* Biến toàn cục phải được khai báo trong file header với từ khóa extern và được định nghĩa trong file source để tránh việc định nghĩa nhiều lần.

**Ví dụ**:

| // Khai báo trong file header extern Motor\_t motorList[10];  // Định nghĩa trong file source Motor\_t motorList[10]; |
| --- |

### **4. Quy tắc sử dụng static và extern**

#### **4.1 Sử dụng static**

* Từ khóa static khi được áp dụng cho biến hoặc hàm sẽ giới hạn phạm vi sử dụng của chúng chỉ trong file source. Điều này giúp bảo vệ các biến và hàm khỏi việc truy cập ngoài ý muốn từ các file khác.
* **Biến static** chỉ được khai báo và định nghĩa trong file source, không được khai báo trong file header để tránh xung đột.

**Ví dụ**:

| // Biến static chỉ được sử dụng trong file source này static uint16\_t motorRunTime[10]; // Biến thời gian hoạt động của động cơ |
| --- |

#### **4.2 Sử dụng extern**

* Từ khóa extern cho phép khai báo các biến và hàm trong file header để chúng có thể được truy cập từ các file khác. Biến extern được khai báo trong file header nhưng phải được định nghĩa trong file source.

**Ví dụ**:

| // Khai báo trong file header extern Motor\_t motorList[10];  // Định nghĩa trong file source Motor\_t motorList[10]; |
| --- |

### **5. Cấu trúc và quy tắc viết comment**

Comment giúp giải thích chức năng của mã, làm cho mã dễ hiểu hơn cho người khác, và giúp bảo trì mã về sau.

#### **5.1 Comment trong file header**

* Comment trong file header bao gồm phần mô tả chi tiết về file (documentation block) và phần mô tả cho các hàm.
* Phần mô tả file giúp lập trình viên hiểu rõ chức năng của file, trong khi phần mô tả hàm giải thích mục đích, các tham số đầu vào và giá trị trả về.

**Ví dụ**:

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file MotorControl.h  \* @brief Khai báo các hàm và cấu trúc liên quan đến điều khiển động cơ  \* @details File này cung cấp giao diện cho việc điều khiển động cơ, bao gồm  \* khởi tạo, đặt tốc độ và dừng động cơ.  \* @author HALA Academy  \* @website https://hala.edu.vn/  \* @version 1.0  \* @date 2024-09-11  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief Khởi tạo hệ thống động cơ.  \* @param motorId ID của động cơ cần khởi tạo.  \* @return void  \*/ void Motor\_Init(uint8\_t motorId); |
| --- |

#### **5.2 Comment trong file source**

* Comment trong file source thường được sử dụng để giải thích chi tiết logic bên trong hàm. Mỗi đoạn mã phức tạp cần có comment để làm rõ chức năng và mục đích của nó.

**Ví dụ**:

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Đặt tốc độ cho động cơ.  \* @param motorId ID của động cơ.  \* @param speed Tốc độ cần đặt (0 - 100%).  \* @return int Trả về 0 nếu thành công, -1 nếu tốc độ không hợp lệ.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ int Motor\_SetSpeed(uint8\_t motorId, uint8\_t speed) {  // Kiểm tra xem tốc độ có nằm trong khoảng hợp lệ không  if (speed > MAX\_MOTOR\_SPEED || speed < MIN\_MOTOR\_SPEED) {  printf("Error: Speed out of range.\n");  return -1; // Trả về lỗi nếu tốc độ không hợp lệ  }  motorList[motorId].motorSpeed = speed;  motorRunTime[motorId] += 1; // Cập nhật thời gian hoạt động của động cơ  printf("Motor %d speed set to %d%%.\n", motorId, speed);  return 0; // Thành công } |
| --- |

### **6. Các ví dụ cụ thể về file header và file source**

#### **6.1 File Header: MotorControl.h**

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file MotorControl.h  \* @brief Khai báo các hàm và cấu trúc liên quan đến điều khiển động cơ  \* @details File này cung cấp giao diện cho việc điều khiển động cơ, bao gồm  \* khởi tạo, đặt tốc độ, dừng động cơ và lấy trạng thái của động cơ.  \* @version 1.0  \* @date 2024-09-11  \* @author HALA Academy  \* @website https://hala.edu.vn/  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #ifndef MOTOR\_CONTROL\_H #define MOTOR\_CONTROL\_H  #include <stdint.h> // Thư viện chuẩn cho kiểu dữ liệu cố định  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Định nghĩa hằng số cho giới hạn tốc độ của động cơ  \* MAX\_MOTOR\_SPEED: Tốc độ tối đa (100%)  \* MIN\_MOTOR\_SPEED: Tốc độ tối thiểu (0%)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ #define MAX\_MOTOR\_SPEED 100U // Tốc độ động cơ tối đa #define MIN\_MOTOR\_SPEED 0U // Tốc độ động cơ tối thiểu  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Định nghĩa kiểu dữ liệu enum cho trạng thái động cơ  \* MOTOR\_OFF: Động cơ tắt  \* MOTOR\_ON: Động cơ bật  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ typedef enum {  MOTOR\_OFF = 0, // Động cơ tắt  MOTOR\_ON = 1 // Động cơ bật } MotorState\_t;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Định nghĩa cấu trúc Motor\_t chứa thông tin về động cơ  \* motorId: ID của động cơ  \* motorSpeed: Tốc độ hiện tại của động cơ  \* motorState: Trạng thái hiện tại của động cơ (ON hoặc OFF)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ typedef struct {  uint8\_t motorId; // ID của động cơ  uint16\_t motorSpeed; // Tốc độ hiện tại của động cơ  MotorState\_t motorState; // Trạng thái hiện tại của động cơ } Motor\_t;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Khai báo biến toàn cục motorList  \* Biến này được sử dụng để lưu thông tin của tất cả các động cơ  \* Mỗi hệ thống có thể quản lý tối đa 10 động cơ  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ extern Motor\_t motorList[10]; // Biến lưu danh sách động cơ  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Khởi tạo hệ thống động cơ  \* @param motorId ID của động cơ cần khởi tạo  \* @return void  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void Motor\_Init(uint8\_t motorId);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Đặt tốc độ cho động cơ  \* @param motorId ID của động cơ cần đặt tốc độ  \* @param speed Tốc độ cần đặt (từ 0% đến 100%)  \* @return int Trả về 0 nếu thành công, -1 nếu tốc độ không hợp lệ  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ int Motor\_SetSpeed(uint8\_t motorId, uint8\_t speed);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Dừng động cơ  \* @param motorId ID của động cơ cần dừng  \* @return void  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void Motor\_Stop(uint8\_t motorId);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Lấy trạng thái hiện tại của động cơ  \* @param motorId ID của động cơ cần kiểm tra trạng thái  \* @return MotorState\_t Trả về trạng thái hiện tại của động cơ  \* MOTOR\_ON hoặc MOTOR\_OFF  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ MotorState\_t Motor\_GetState(uint8\_t motorId);  #endif // MOTOR\_CONTROL\_H |
| --- |

#### **6.2 File Source: MotorControl.c**

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file MotorControl.c  \* @brief Định nghĩa các hàm điều khiển động cơ  \* @details File này chứa phần định nghĩa của các hàm điều khiển động cơ, bao gồm  \* khởi tạo, đặt tốc độ, dừng động cơ, và lấy trạng thái động cơ.  \* @version 1.0  \* @date 2024-09-11  \* @author HALA Academy  \* @website https://hala.edu.vn/  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "MotorControl.h" // File header chứa các khai báo liên quan #include <stdio.h> // Thư viện chuẩn để sử dụng hàm printf  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Định nghĩa biến motorList là danh sách các động cơ trong hệ thống  \* Hệ thống hỗ trợ tối đa 10 động cơ, với mỗi động cơ lưu trong motorList  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ Motor\_t motorList[10]; // Biến toàn cục lưu thông tin về các động cơ  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Biến static motorRunTime lưu thời gian hoạt động của từng động cơ  \* Biến này chỉ được sử dụng trong phạm vi file source này (local scope)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ static uint16\_t motorRunTime[10]; // Thời gian hoạt động của mỗi động cơ  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Khởi tạo động cơ  \* @details Hàm này đặt trạng thái động cơ về OFF và tốc độ ban đầu là 0.  \* Biến motorRunTime được đặt về 0 cho mỗi động cơ.  \* @param motorId ID của động cơ cần khởi tạo  \* @return void  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void Motor\_Init(uint8\_t motorId) {  motorList[motorId].motorId = motorId; // Gán ID cho động cơ  motorList[motorId].motorSpeed = 0U; // Đặt tốc độ ban đầu là 0  motorList[motorId].motorState = MOTOR\_OFF; // Đặt trạng thái là OFF  motorRunTime[motorId] = 0U; // Thời gian chạy về 0  printf("Motor %d initialized.\n", motorId); // In ra thông báo khởi tạo }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Đặt tốc độ cho động cơ  \* @details Hàm này kiểm tra xem tốc độ truyền vào có hợp lệ không (trong khoảng 0-100%).  \* Nếu hợp lệ, tốc độ của động cơ được cập nhật và thời gian hoạt động được tăng lên.  \* @param motorId ID của động cơ cần đặt tốc độ  \* @param speed Tốc độ cần đặt (từ 0% đến 100%)  \* @return int Trả về 0 nếu thành công, -1 nếu tốc độ không hợp lệ  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ int Motor\_SetSpeed(uint8\_t motorId, uint8\_t speed) {  // Kiểm tra xem tốc độ có nằm trong khoảng hợp lệ không  if (speed > MAX\_MOTOR\_SPEED || speed < MIN\_MOTOR\_SPEED) {  printf("Error: Speed out of range.\n"); // In ra lỗi nếu không hợp lệ  return -1; // Trả về -1 nếu không hợp lệ  }   motorList[motorId].motorSpeed = speed; // Cập nhật tốc độ động cơ  motorList[motorId].motorState = MOTOR\_ON; // Đặt trạng thái động cơ là ON  motorRunTime[motorId] += 1; // Tăng thời gian hoạt động của động cơ  printf("Motor %d speed set to %d%%.\n", motorId, speed); // In ra tốc độ mới  return 0; // Trả về 0 nếu thành công }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Dừng động cơ  \* @details Hàm này dừng động cơ bằng cách đặt tốc độ về 0 và trạng thái về OFF.  \* @param motorId ID của động cơ cần dừng  \* @return void  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void Motor\_Stop(uint8\_t motorId) {  motorList[motorId].motorSpeed = 0U; // Đặt tốc độ về 0  motorList[motorId].motorState = MOTOR\_OFF; // Đặt trạng thái về OFF  printf("Motor %d stopped.\n", motorId); // In ra thông báo dừng động cơ }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @brief Lấy trạng thái hiện tại của động cơ  \* @details Hàm này trả về trạng thái hiện tại của động cơ (MOTOR\_ON hoặc MOTOR\_OFF).  \* @param motorId ID của động cơ cần kiểm tra  \* @return MotorState\_t Trả về trạng thái của động cơ (ON hoặc OFF)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ MotorState\_t Motor\_GetState(uint8\_t motorId) {  return motorList[motorId].motorState; // Trả về trạng thái hiện tại của động cơ } |
| --- |