

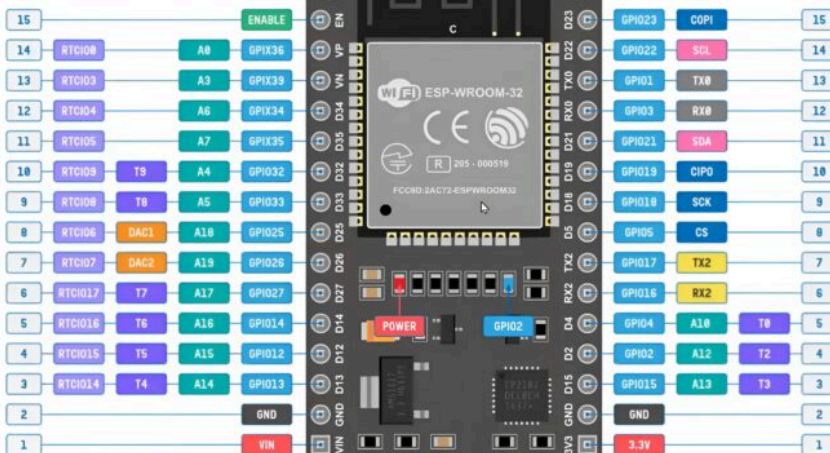
# KIT ESP32 3



## Board Esp32 dev kit v1

PHYSICAL PIN	POSITIVE SUPPLY
CONTROL PINS	GROUND SUPPLY
GPIO PINS	ADC INPUTS
DAC OUTPUTS	SPI PINS
TOUCH INPUTS	UART PINS
I2C PINS	EXCLUDED PINS

- GPIO pins 34, 35, 36 and 39 are input only.
- TX0 and RX0 (Serial0) are used for serial programming.
- TX2 and RX2 can be accessed as Serial2.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and Interrupts.
- Built-in LED is connected to GPIO2.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.



Copyright by Điện thông minh E-smart

## Câu hỏi:

Thầy cho em xin phép hỏi là khi mà chúng ta thiết lập chạy code, nếu mà ta đưa các cái thiết bị ngoại vi vào các cái đầu ra này thì mức logic đầu ra của nó sẽ như thế nào? Nếu mà mình đưa cái mức logic High vào chân 12 thì nó sẽ gây ảnh hưởng đến cái hoạt động của con vi điều khiển này, còn nếu mà ta cứ để cố định mức logic 0 thì nó không ảnh hưởng gì đúng không thầy?

## Trả lời:

**1. Khái niệm chân Strapping:** đây là những chân đặc biệt đóng vai trò như các công tắc cấu hình phần cứng, giúp chip ESP32 tự xác định chế độ vận hành ngay tại thời điểm cực ngắn khi vừa được cấp nguồn hoặc nhấn nút Reset.

**2. Vai trò của chân GPIO 12:** chân này chịu trách nhiệm

điều khiển bộ điều tiết điện áp cấp cho chip nhớ Flash nội bộ, nơi lưu trữ toàn bộ mã nguồn chương trình mà bạn đã nạp vào board mạch.

**3. Quy định mức logic:** theo thiết kế của nhà sản xuất, nếu lúc khởi động chân 12 ở mức logic 0 thì chip sẽ cấp điện áp 3.3V cho Flash, còn nếu chân này ở mức logic 1 thì chip sẽ chuyển sang cấp điện áp 1.8V cho Flash.

**4. Nguyên nhân gây lỗi treo chip:** hầu hết các bộ kit ESP32 DevKit V1 hiện nay đều dùng chip nhớ Flash 3.3V, do đó nếu bạn đưa mức High vào chân 12 lúc boot, chip sẽ cấp thiếu điện áp khiến bộ nhớ không thể hoạt động và gây ra lỗi treo hệ thống.

**5. Sự an toàn khi ở mức logic 0:** nhận định của bạn là hoàn toàn chính xác vì khi giữ chân 12 ở mức thấp lúc khởi động, chip sẽ luôn nhận đúng điện áp tiêu chuẩn và vận hành ổn định mà không gặp lỗi đọc bộ nhớ Flash.

**6. Thời điểm ảnh hưởng của mức logic:** bạn cần lưu ý rằng trạng thái ràng buộc này chỉ có ý nghĩa trong vài mili giây đầu tiên khi Boot, sau khi chip đã khởi động thành công và chạy vào hàm setup, bạn có thể dùng chân 12 như một công tắc bình thường để điều khiển ngoại vi.

**7. Giải pháp thiết kế mạch:** để đảm bảo an toàn, bạn nên tránh đưa các cảm biến tự động xuất mức High vào chân 12 khi vừa có điện, thay vào đó hãy sử dụng nút nhấn thường hoặc gắn thêm điện trở kéo xuống GND để cố định mức 0 lúc vừa bật nguồn.

**Kết luận:** việc nắm rõ đặc tính của các chân Strapping như GPIO 12 là chìa khóa để thiết kế mạch ESP32 ổn định, giúp tránh được các lỗi phần cứng khó hiểu ngay từ giai đoạn khởi động và đảm bảo hệ thống luôn vận hành đúng ý của người lập trình.

...

## Câu hỏi:

Ngoài gpio 12, còn các chân strapping khác không?

## Trả lời:

Ngoài chân GPIO 12 mà chúng ta vừa thảo luận, chip ESP32 còn có một vài chân Strapping khác cũng cực kỳ quan trọng. Các chân này quyết định việc chip sẽ vào chế độ nạp code (Download Mode) hay chế độ chạy chương trình (Boot Mode), cũng như các thiết lập về tần số và điện áp khởi động.

**1. GPIO 0:** đây là chân quan trọng nhất để quyết định chế độ khởi động, nếu tại thời điểm reset chân này ở mức Low (0) thì ESP32 sẽ vào chế độ nạp code, còn nếu ở mức High (1) thì chip sẽ chạy chương trình bình thường từ bộ nhớ Flash.

**2. GPIO 2:** chân này thường phải ở mức Low (0) hoặc để trống trong lúc khởi động để chip có thể vào chế độ nạp code thành công, nếu chân này bị kéo lên mức High quá cao có thể khiến quá trình nạp chương trình bị lỗi.

**3. GPIO 5:** đây là chân điều khiển tín hiệu cầu hình cho các chân xuất tín hiệu của chip (SDIO Slave), thông thường bạn nên để chân này ở mức High (1) lúc khởi động để đảm bảo các thiết lập mặc định của hệ thống được thực thi đúng.

**4. GPIO 15:** chân này điều khiển việc bật hoặc tắt các thông báo gỡ lỗi (debug log) từ chip ra cổng Serial khi vừa mới khởi động, nếu chân này ở mức High (1) thì chip sẽ xuất các dòng log khởi động giúp bạn dõi theo diễn biến của hệ thống.

**5. Sự khác biệt giữa các dòng chip:** bạn cần lưu ý rằng

tùy vào phiên bản ESP32 (như bản S2, S3 hay C3) mà số lượng và chức năng của các chân Strapping có thể thay đổi nhẹ, nhưng đối với bộ kit DevKit V1 của bạn thì 5 chân (0, 2, 5, 12, 15) là những chân cốt lõi nhất.

**6. Rủi ro khi đấu ngoại vi vào các chân này:** giống như chân 12, nếu bạn đấu các module như Relay hoặc cảm biến mà chúng tự động kéo các chân này về mức logic ngược với mặc định lúc vừa bật nguồn, ESP32 sẽ không thể khởi động hoặc liên tục rơi vào chế độ nạp code.

**7. Cách sử dụng an toàn:** lời khuyên tốt nhất là bạn nên ưu tiên sử dụng các chân GPIO khác không nằm trong danh sách Strapping cho các thiết bị ngoại vi quan trọng, hoặc nếu bắt buộc phải dùng thì cần thiết kế thêm trở kéo (pull-up/pull-down) để đảm bảo mức logic đúng tại thời điểm Boot.

**Kết luận:** việc quản lý tốt trạng thái logic của các chân GPIO 0, 2, 5, 12 và 15 khi khởi động là yêu cầu quyết định giúp bo mạch ESP32 vận hành trơn tru, giúp bạn tránh được những lỗi “chip không chạy” hoặc “không nạp được code” rất phổ biến trong quá trình phát triển dự án.

Bạn có muốn mình liệt kê danh sách các chân GPIO “vùng xanh” hoàn toàn không có chức năng Strapping để bạn thoải mái đấu nối mà không cần tính toán logic không?

...

**Câu hỏi:**

Giới thích Strapping là gì và liệt kê danh sách các chân GPIO “vùng xanh” hoàn toàn không có chức năng Strapping để bạn thoải mái đấu nối mà không cần tính toán logic không?

## Trò lời:

**1. Định nghĩa Strapping:** strapping được hiểu là các chân “ràng buộc” cụ thể, chúng đóng vai trò như các công tắc vật lý giúp chip ESP32 tự kiểm tra trạng thái logic (0 hoặc 1) ngay tại thời điểm vài mili giây khi vừa nhận Reset hoặc cấp nguồn để quyết định chế độ khởi động.

**2. Cơ chế hoạt động:** khi chip vừa có điện, bộ điều khiển nội bộ sẽ quét qua các chân này, nếu mức logic không đúng như mặc định (do thiết bị ngoại vi kéo lên hoặc kéo xuống), chip sẽ đi vào sai chế độ như liên tục chờ nạp code hoặc cấp sai điện áp cho bộ nhớ Flash.

**3. Các chân cần tránh:** như đã thảo luận, các chân GPIO 0, 2, 5, 12, và 15 là nhóm Strapping chính mà bạn cần cực kỳ cẩn trọng khi đưa nối các thiết bị có khả năng làm thay đổi mức logic lúc khởi động.

**4. Danh sách chân vùng xanh an toàn:** đây là các chân GPIO hoàn toàn không dính dáng đến quá trình khởi động, bao gồm GPIO 4, 13, 14, 16, 17, 18, 19, 21, 22, 23, 25, 26, 27, 32 và 33.

**5. Nhóm chân chỉ nhận đầu vào:** bạn cần lưu ý thêm nhóm chân GPIO 34, 35, 36 (VP) và 39 (VN) cũng là các chân an toàn nhưng chúng chỉ có thể làm đầu vào (Input-only) và không có điện trở kéo lên nội bộ, rất thích hợp để đọc cảm biến.

**6. Lợi ích của vùng xanh:** khi sử dụng các chân này, bạn có thể thoải mái kết nối với Relay, LED, cảm biến hay bất kỳ linh kiện nào mà không bao giờ phải lo lắng về việc chip bị treo hoặc không nạp được code do xung đột mức logic.

**7. Thứ tự ưu tiên thiết kế:** lời khuyên tốt nhất là bạn

nên dùng hết các chân “vùng xanh” này trước cho các module điếu khi quan trọng, chỉ khi nào thiếu chân mới bắt đầu tính toán đến việc tận dụng các chân Strapping một cách cẩn thận.

**Kết luận:** việc ưu tiên sử dụng các chân không có chức năng Strapping và nôm rỗ danh sách “vùng xanh” sẽ giúp sơ đồ mạch của bạn trở nên chuyên nghiệp, ổn định và loại bỏ hoàn toàn các rủi ro phần cứng ngay từ bước phác thảo.

...

### **Câu hỏi:**

Khi arduino xuất file đã là chuẩn 01 thì chip uart chỉ là người đưa thư, thì tại sao gọi nó là chip usb to uart?

### **Trả lời:**

Bạn đặt một câu hỏi rất hay và mang tính kỹ thuật chuyên sâu! Tuy nhiên, có một sự nhầm lẫn nhỏ về “ngôn ngữ” mà các sợi dây dẫn sử dụng. Để làm rõ tại sao nó được gọi là chip USB-to-UART dù nó chỉ là “người đưa thư”, mình sẽ giải thích qua 7 ý chính:

**1. Sự khác biệt vật lý:** sợi cáp USB có 4 dây (Nguồn, GND, D+, D-) và truyền tín hiệu theo chuẩn vi sai tốc độ cao, trong khi chân chip ESP32 chỉ hiểu chuẩn UART gồm 2 dây (TX, RX) với mức điện áp 0V và 3.3V.

**2. Nhiệm vụ chuyển đổi giao thức:** chip USB-to-UART không chỉ đưa thư mà còn là một “phiên dịch viên” điện tử, nó bóc tách các gói dữ liệu phức tạp từ chuẩn USB và chuyển chúng thành các xung nhịp đơn giản của chuẩn UART để ESP32 có thể tiếp nhận.

**3. Định dạng file hex:** đúng là Arduino IDE biên dịch ra

file Hex (hoặc Binary), nhưng “Hex” chỉ là nội dung của bức thư, còn “USB” hay “UART” là phương tiện vận chuyển (giống như việc bạn gửi thư bằng máy bay hay bằng xe máy vậy).

**4. Lý do không gọi là USB-to-ESP32:** sẽ dễ gọi là USB-to-UART vì con chip này có thể dùng cho bất cứ vi điều khiển nào có cổng UART (như Arduino Nano, STM32, ESP8266) chứ không chỉ riêng ESP32.

**5. Tiến trình nạp code:** khi bạn nhấn “Upload”, máy tính sẽ gửi file Hex đó thông qua Driver USB, chip chuyển đổi sẽ “hứng” dữ liệu này rồi đẩy vào chân RX của ESP32 theo đúng tốc độ baudrate mà bạn đã chọn (ví dụ 115200).

**6. Vai trò nạp tự động:** con chip này còn làm thêm một việc “người đưa thư” thông minh là điều khiển các chân Reset (EN) và Boot (GPIO 0) của ESP32 để bất chip dừng chương trình cũ và chuẩn bị ghi chương trình mới vào Flash.

**7. Tính thông dụng:** nếu không có con chip này, bạn sẽ phải dùng một bộ nạp rời bên ngoài (như mạch nạp FTDI hoặc CP2102 rời) để nối dây vào các chân TX/RX của board mới có thể nạp code được.

**Kết luận:** tóm lại, tên gọi USB-to-UART phản ánh đúng chức năng “đổi cầu” tín hiệu từ chuẩn USB sang chuẩn UART, giúp máy tính và vi điều khiển có thể “nói chuyện” được với nhau dù chúng sử dụng hai loại điện áp và giao thức hoàn toàn khác biệt.

...

**Câu hỏi:**

Vậy gọi Chip USB-to-UART bridge là đây để nh?

## Trò lời:

Chính xác, gọi là Chip USB-to-UART Bridge là cách gọi đầy đủ, chuyên nghiệp và mô tả đúng nhất bản chất kỹ thuật của linh kiện này trên board ESP32.

**1. Ý nghĩa của từ Bridge:** từ “Bridge” (Cầu nối) không định đây là một thực thể trung gian, kết nối hai giao thức truyền thông khác nhau là USB (phía máy tính) và UART (phía vi điều khiển) mà không can thiệp vào nội dung dữ liệu.

**2. Chức năng chuyển đổi vật lý:** nó thực hiện nhiệm vụ quan trọng là chuyển đổi mức điện áp và cách thức đóng gói dữ liệu từ chuẩn vi sai của USB sang chuẩn nối tiếp đơn giản TX/RX của UART mà chip ESP32 có thể tiếp nhận.

**3. Hiện diện trong hệ thống:** trong sơ đồ khối của một bộ Kit ESP32 DevKit V1, chip Bridge này nằm tách biệt hoàn toàn với chip xử lý trung tâm, đóng vai trò như một “cầu nối” duy nhất để nạp code và giao tiếp Serial.

**4. Vai trò “người đưa thư” chuyên trách:** đúng như bạn nhận định, nó chỉ chuyển tải “bức thư” (file Hex/Bin) từ máy tính đến vi điều khiển, nhưng nó phải hoạt động cực kỳ chính xác về mặt thời gian (baudrate) để dữ liệu không bị thất lạc.

**5. Việc đăng ký thiết bị với hệ điều hành:** khi bạn cài Driver, máy tính sẽ nhận diện thiết bị này dưới tên đầy đủ là “USB-to-UART Bridge”, từ đó tạo ra một cổng COM ảo (Virtual COM Port) để các phần mềm như Arduino IDE có thể nhìn thấy và truyền dữ liệu.

**6. Tích hợp mạch điều khiển nạp:** một bộ USB-to-UART Bridge đầy đủ trên ESP32 còn bao gồm các chân điều khiển phụ (như DTR và RTS) để phải hợp với mạch nạp tự động, giúp khối động lại chip và đưa chip vào chế độ

Bootloader mà không cần thao tác tay.

**7. Tính ứng dụng rộng rãi:** việc hiểu rõ tên gọi này giúp bạn phân biệt được với các loại Bridge khác như USB-to-SPI hay USB-to-I2C, từ đó giúp việc tra cứu tài liệu kỹ thuật (Datasheet) và sửa lỗi kết nối trở nên dễ dàng hơn.

**Kết luận:** sử dụng cụm từ Chip USB-to-UART Bridge trong tài liệu sẽ giúp người đọc hiểu rằng đây là một giải pháp kết nối phần cứng hoàn chỉnh, bao gồm cả việc chuyển đổi giao thức lên việc tạo cầu nối liên lạc bền vững cho hệ thống vi điều khiển.