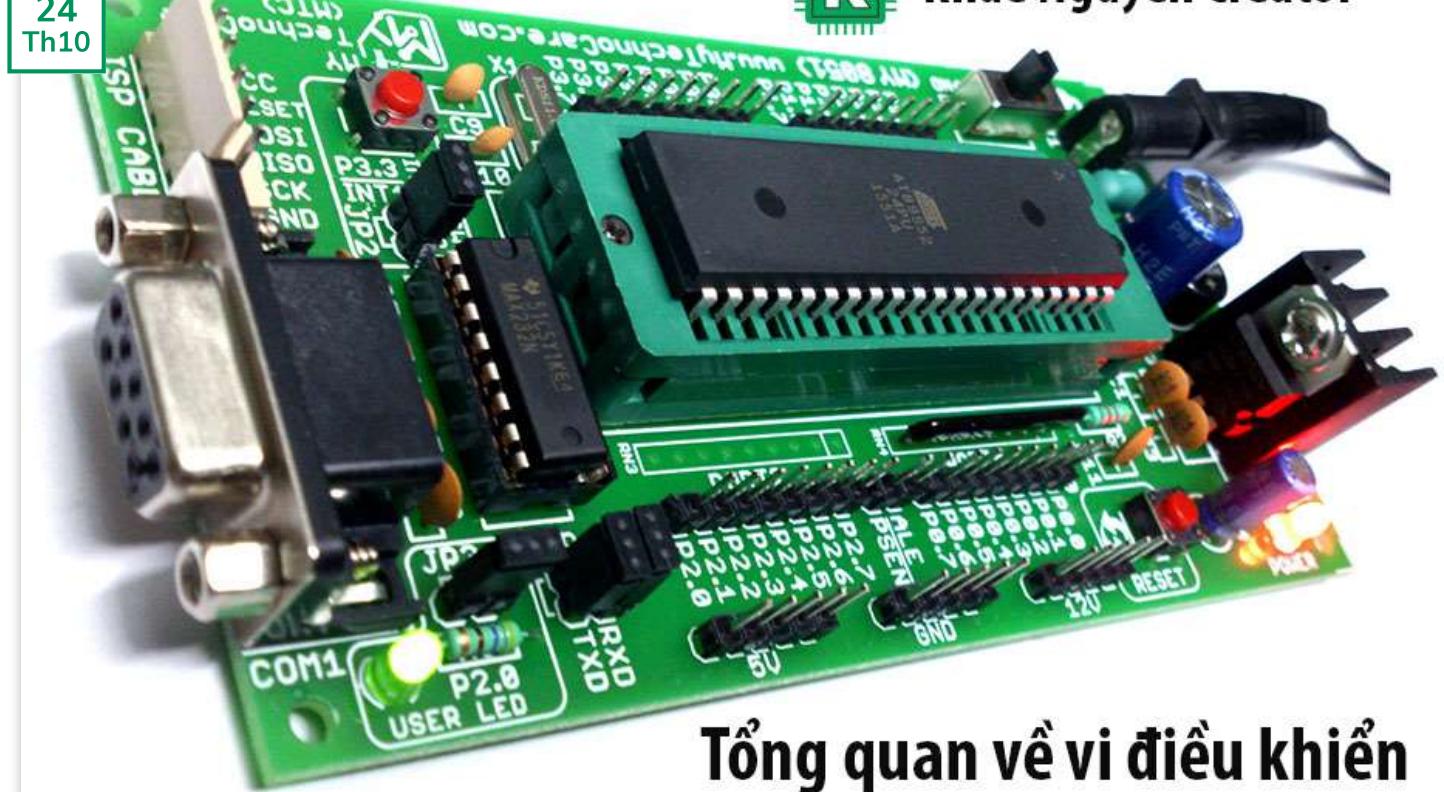


**KIẾN TRÚC VI ĐIỀU KHIỂN**

# Tổng quan về vi điều khiển, cấu tạo và cách hoạt động

POSTED ON 24/10/2021 BY KHUÊ NGUYỄN

24  
Th10**Khuê Nguyễn Creator**

## Tổng quan về vi điều khiển

Trong lập trình nhúng, kiến trúc vi điều khiển là rất quan trọng. Nếu nói **lập trình** là tạo ra trí thông minh cho máy tính, thì việc hiểu rõ cấu trúc, cấu tạo của máy tính sẽ giúp chúng ta hiểu cách vận hành của chúng. Từ đó điều khiển chúng một cách dễ dàng và mượt mà hơn.

Đây là những kiến thức bắt buộc phải học đối với mỗi một kĩ sư nhúng. Vậy, cùng tìm hiểu nhé!

## Mục Lục



1. Vi điều khiển là gì ?
2. Các họ vi điều khiển
  - 2.1. Họ vi điều khiển Atmel
  - 2.2. Họ vi điều khiển STMicroelectronics
  - 2.3. Họ vi điều khiển Microchip
  - 2.4. Các dòng vi điều khiển khác
3. Phân loại vi điều khiển
  - 3.1. Phân loại theo độ dài thanh ghi
  - 3.2. Phân loại theo kiến trúc CISC và RISC
  - 3.3. Kiến trúc Harvard và kiến trúc Von-Neumann
4. Tại sao chúng ta hay nhầm lẫn giữa vi điều khiển và vi xử lý
  - 4.1. Điểm giống nhau
  - 4.2. Điểm khác biệt
5. Cấu trúc tổng quan của vi điều khiển
  - 5.1. CPU hay VI xử lý
  - 5.2. Oscillator Circuit
  - 5.3. Memory – Bộ nhớ
  - 5.4. Timer/counter
  - 5.5. Các ngoại vi của vi điều khiển
    - 5.5.1. I/O Ports – Input/output
    - 5.5.2. Các chuẩn giao tiếp
    - 5.5.3. Bộ chuyển đổi analog sang digital (ADC)
    - 5.5.4. Bộ chuyển đổi Digital sang Analog (DAC)
  - 5.6. Interrupt control hay quản lý sự kiện
  - 5.7. Special functioning block
6. Tiếp cận với vi điều khiển như thế nào?
  - 6.1. Chọn dòng vi điều khiển nào?
  - 6.2. Lập trình cho dòng vi điều khiển đó
  - 6.3. Nạp chương trình
  - 6.4. Debug chương trình
7. Ưu và nhược điểm của vi điều khiển
  - 7.1. Ưu điểm của vi điều khiển

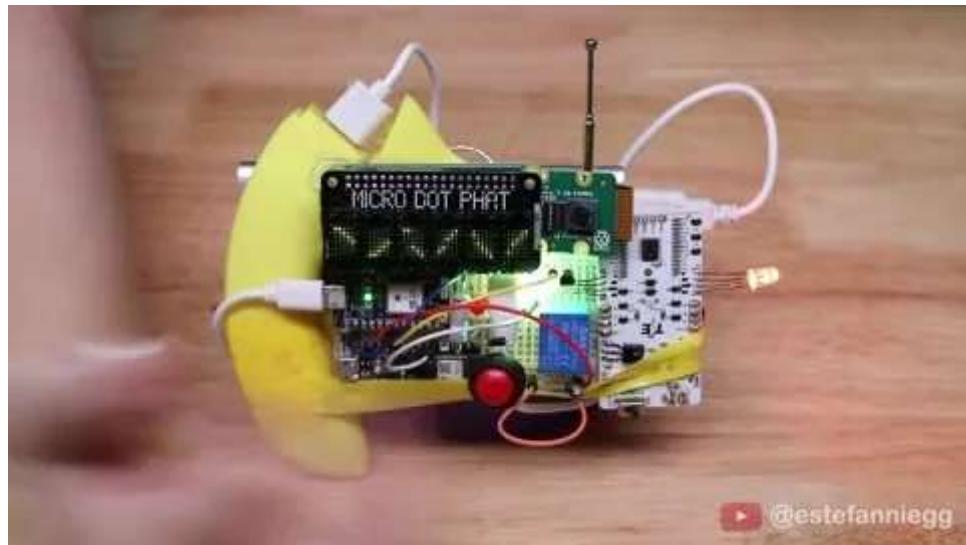
- 7.2. Nhược điểm của vi điều khiển
- 8. Ứng dụng của vi điều khiển
- 9. Kết
  - 9.1. Related posts:

## Vi điều khiển là gì ?

**Vi điều khiển** là một máy tính được tích hợp trên một chip, nó thường được sử dụng để điều khiển các thiết bị điện tử. Vi điều khiển, thực chất, là một hệ thống bao gồm một vi xử lý có hiệu suất đủ dùng và giá thành thấp (khác với các bộ vi xử lý đa năng dùng trong máy tính) kết hợp với các khối ngoại vi như bộ nhớ, các module vào/ra, các module biến đổi số sang tương tự và tương tự sang số,...

Vi điều khiển thường được sử dụng để xây dựng các hệ thống nhúng. Nó cũng được sử dụng trong các thiết bị điện, điện tử như máy giặt, lò vi sóng, điện thoại, đầu đọc DVD, thiết bị đa phương tiện hay dây chuyền sản xuất tự động,...

Theo [wikipedia](#).



## Các họ vi điều khiển

### Họ vi điều khiển Atmel

Đây là một dòng đã quá quen thuộc khi các bạn học vi điều khiển trên ghế nhà trường, điển hình của nó là họ 8051. Ngoài ra còn có các dòng như sau:

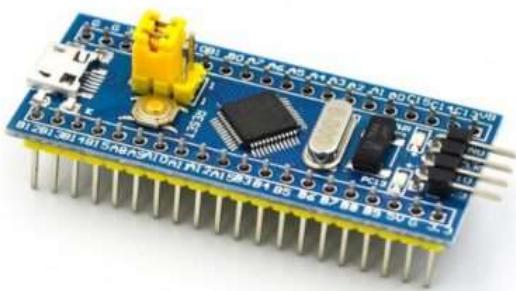
- Dòng 8051 (8031, 8051, 8751, 8951, 8032, 8052, 8752, 8952)
- Dòng Atmel AT91 (Kiến trúc ARM THUMB)
- Dòng AT90, Tiny & Mega – AVR (Atmel Norway design)
- Dòng Atmel AT89 (Kiến trúc Intel 8051/MCS51)
- Dòng MARC4



## Họ vi điều khiển STMicroelectronics

Đây là dòng chip chủ đạo trong các bài học của mình, đại diện chính là dòng STM32 huyền thoại

- ST 62
- ST7
- STM8
- STM32 (Cortex-Mx)



## Hộ vi điều khiển Microchip

Quá quen thuộc với các dòng PIC huyền thoại. VD:

- PIC 8-bit (xử lý dữ liệu 8-bit, 8-bit data bus)
  - Từ lệnh dài 12-bit (Base-line): PIC10F, PIC12F và một vài PIC16F
  - Từ lệnh dài 14-bit (Mid-Range và Enhance Mid-Range): PIC16Fxxx, PIC16F1xxx
  - Từ lệnh dài 16-bit (High Performance): PIC18F
- PIC 16-bit (xử lý dữ liệu 16-bit)
  - PIC điều khiển động cơ: dsPIC30F
  - PIC có DSC: dsPIC33F
  - Phổ thông: PIC24F, PIC24E, PIC24H
- PIC 32-bit (xử lý dữ liệu 32-bit): PIC32MX

## Các dòng vi điều khiển khác

Ngoài ra còn có các dòng ít gặp của các hãng khác như:

- Họ vi điều khiển Cypress MicroSystems
- Họ vi điều khiển AMCC (Applied Micro Circuits Corporation)
- Họ vi điều khiển Freescale Semiconductor.
- Họ vi điều khiển Intel
- Họ vi điều khiển National Semiconductor
- Họ vi điều khiển Philips Semiconductors

## Phân loại vi điều khiển

### Phân loại theo độ dài thanh ghi

Dựa vào độ dài của các thanh ghi và các lệnh của VĐK mà người ta chia ra các loại vi điều khiển 8 bit, 16 bit hay 32 bit ...

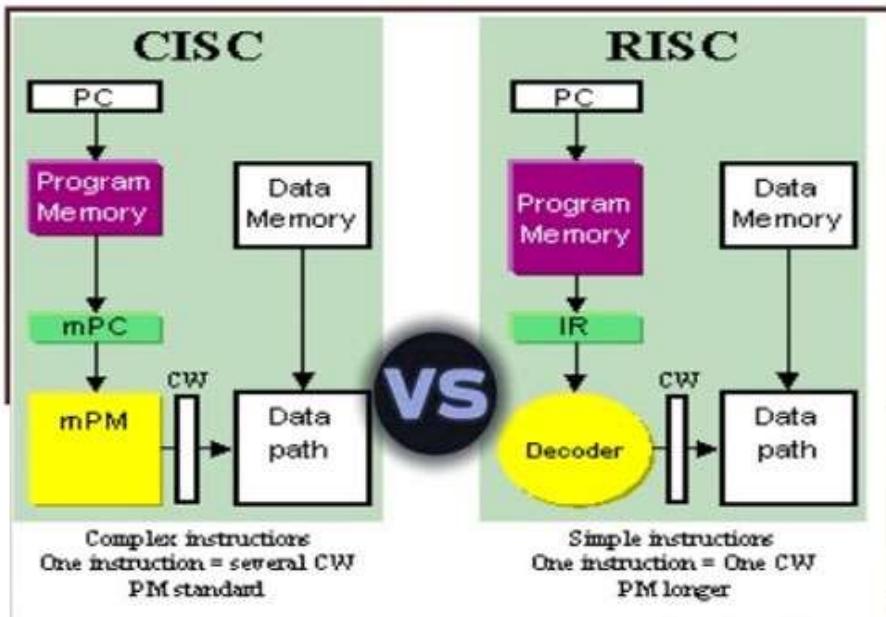
Các loại VĐK 16 bit do có độ dài lệnh lớn hơn nên các tập lệnh cũng nhiều hơn, phong phú hơn. Tuy nhiên bất cứ chương trình nào viết bằng VĐK 16 bit chúng ta đều có thể viết trên vi điều khiển 8 bit với chương trình thích hợp.

### Phân loại theo kiến trúc CISC và RISC

Vi điều khiển CISC là vi điều khiển có tập lệnh phức tạp. Các VĐK này có một số lượng lớn các lệnh nên giúp cho người lập trình có thể linh hoạt và dễ dàng hơn khi viết chương trình.

Vi điều khiển RISC là vi điều khiển có tập lệnh đơn giản. Chúng có một số lượng nhỏ các lệnh đơn giản. Do đó, chúng đòi hỏi phần cứng ít hơn, giá thành thấp hơn, và nhanh hơn so với CISC. Tuy nhiên nó đòi hỏi người lập trình phải viết các chương trình phức tạp hơn, nhiều lệnh hơn.

## CISC & RISC PROCESSORS



### Kiến trúc Harvard và kiến trúc Von-Neumann

Kiến trúc Harvard sử dụng bộ nhớ riêng biệt cho chương trình và dữ liệu. Bus địa chỉ và bus dữ liệu độc lập với nhau nên quá trình truyền nhận dữ liệu đơn giản hơn. Kiến trúc Von-Neumann sử dụng chung bộ nhớ cho chương trình và dữ liệu. Điều này làm cho VĐK gọn nhẹ hơn, giá thành rẻ hơn.

### Tại sao chúng ta hay nhầm lẫn giữa vi điều khiển và vi xử lý

Chúng ta thường bị nhầm lẫn giữa vi điều khiển và vi xử lý. Vậy rốt cuộc chúng giống và khác nhau gì?

#### Điểm giống nhau

Vi điều khiển và vi xử lý đều xử lý thông tin điều khiển sự hoạt động của máy tính hoặc **mạch điện**.

Chúng có kích thước và hình dáng khá giống nhau.

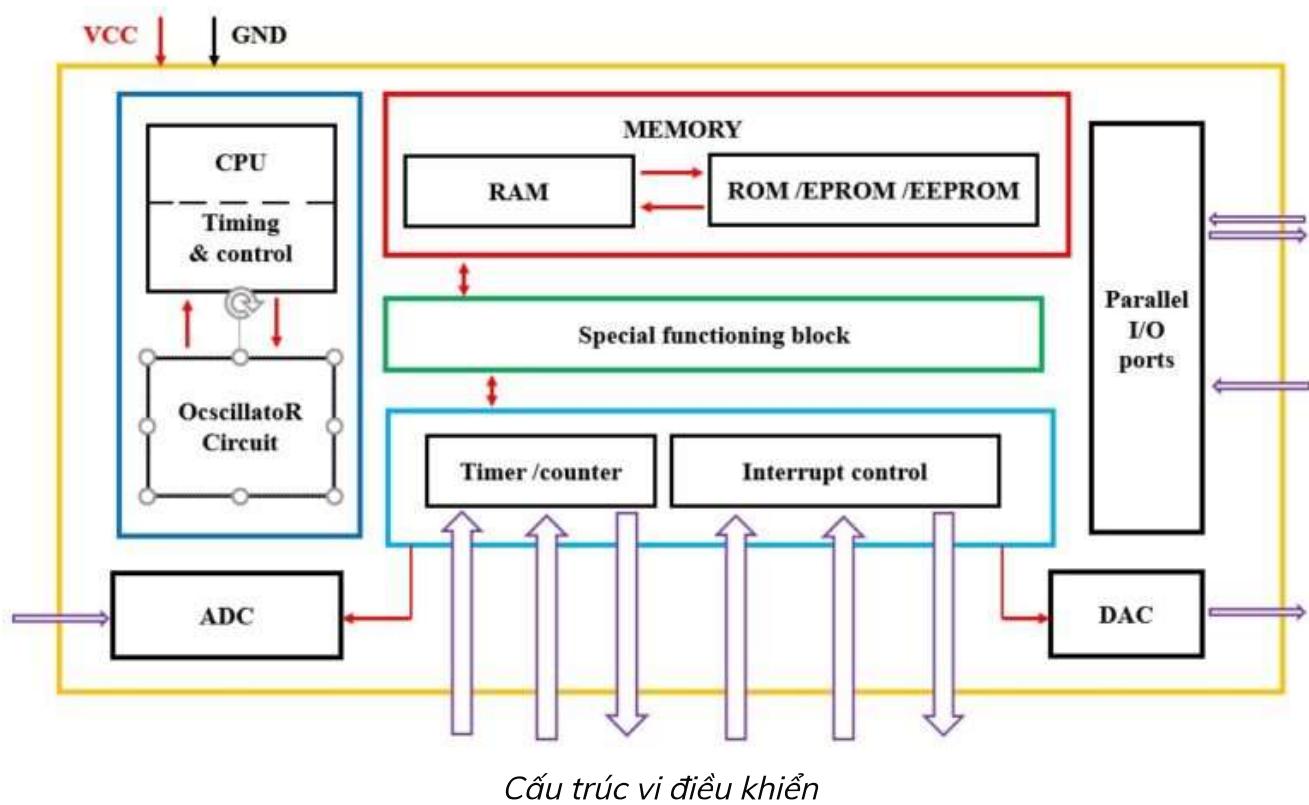
Vì vậy nên sẽ rất dễ nhầm lẫn 2 khái niệm này với nhau. Vậy chúng khác nhau như thế nào?

## Điểm khác biệt

Nếu ví vi điều khiển như một con người thì vi xử lý chính là bộ não.

- Vi điều khiển có thể hoạt động độc lập, tương tác với thế giới bên ngoài bằng các ngoại vi như ADC, các chân IO, các chuẩn giao tiếp I2C, SPI,... Còn vi xử lý chỉ có thể tiếp nhận thông tin, phân tích và điều khiển qua các bus dữ liệu.
- Vi điều khiển là sự tích hợp của vi xử lý và nhiều các thành phần khác nhau nữa như bộ nhớ, ngoại vi, bộ định thời,... Đối với vi xử lý, để hoạt động được chúng cần có các bộ nhớ ngoài như RAM, ổ cứng,... các bộ định thời như RTC...
- Lập trình vi điều khiển thường được sử dụng để làm các thiết bị tự động, còn lập trình vi xử lý thường để làm các hệ điều hành dùng trong máy tính hoặc các sản phẩm tương tự máy tính. Tuy vậy vi xử lý cũng có thể sử dụng trong các thiết bị như máy tính nhúng, có thể kể đến như Ras Pi, Jetson...
- Vi xử lý sẽ quan trọng phần hiệu năng làm việc, vi xử lý càng có hiệu năng tốt thì càng mạnh mẽ, còn vi điều khiển sẽ quan trọng phần tối ưu giữa công xuất và hiệu năng, bởi các ứng dụng nhúng đòi hỏi không cần tốc độ làm việc quá cao mà sẽ quan tâm tới việc tiết kiệm năng lượng và ổn định.

## Cấu trúc tổng quan của vi điều khiển



## CPU hay Vi xử lý

**CPU (Center Programing Unit)** hay bộ xử lý trung tâm là bộ não của vi điều khiển. CPU chịu trách nhiệm nạp lệnh, giải mã và thực thi. Tất cả những hành vi của vi điều khiển đều là do CPU điều khiển.

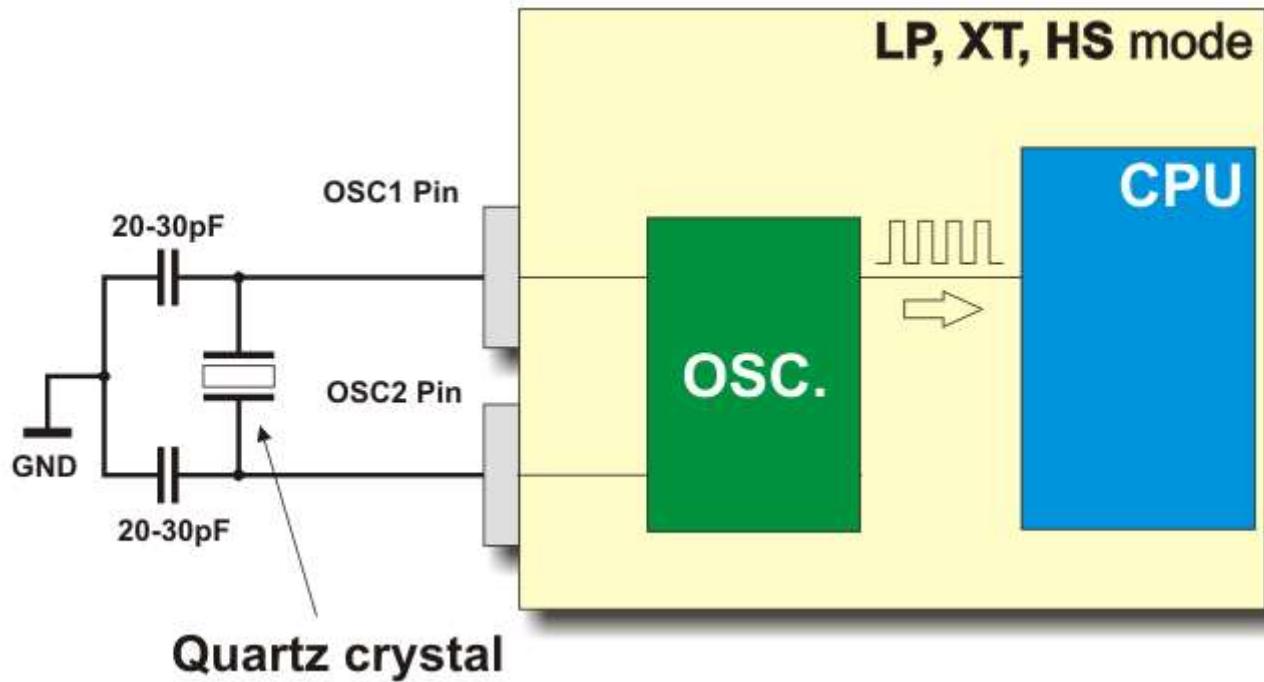
Chúng giao tiếp với các phần khác trong vi điều khiển thông qua hệ thống Bus.



## Oscillator Circuit

Nếu CPU là bộ não thì Oscillator Circuit hay còn gọi là Clock được coi là trái tim của vi điều khiển. Để mọi thứ có thể hoạt động, bắt buộc chúng ta phải cấp xung, trái tim hoạt động mới có thể bơm máu cho toàn bộ cơ thể hoạt động được.

Chúng ta thường nghe quảng cáo dòng vi xử lý có tốc độ bao nhiêu Ghz gì gì đó, chính là tốc độ Clock mà vi xử lý đó có thể đáp ứng được, tốc độ xung càng cao thì tốc độ xử lý của CPU cũng tăng lên. đương nhiên mọi thứ đều có giới hạn của nó.



## Memory – Bộ nhớ

Bộ nhớ có thể coi là một phần không thể thiếu, chúng là nơi lưu trữ chương trình nạp lên hoặc dùng làm nơi chứa các thông tin tức thời mà CPU cần dùng tới. Có 2 kiểu bộ nhớ cơ bản:

- RAM (Random access memory) là bộ nhớ lưu các dữ liệu mà CPU cần dùng để tính toán, đưa ra quyết định, chúng sẽ bị xóa khi mất điện
- ROM/EPROM/EEPROM hoặc Flash: là bộ nhớ lưu trữ chương trình hay trí khôn của vi điều khiển, chúng được ghi khi chúng ta nạp chương trình vào vi điều

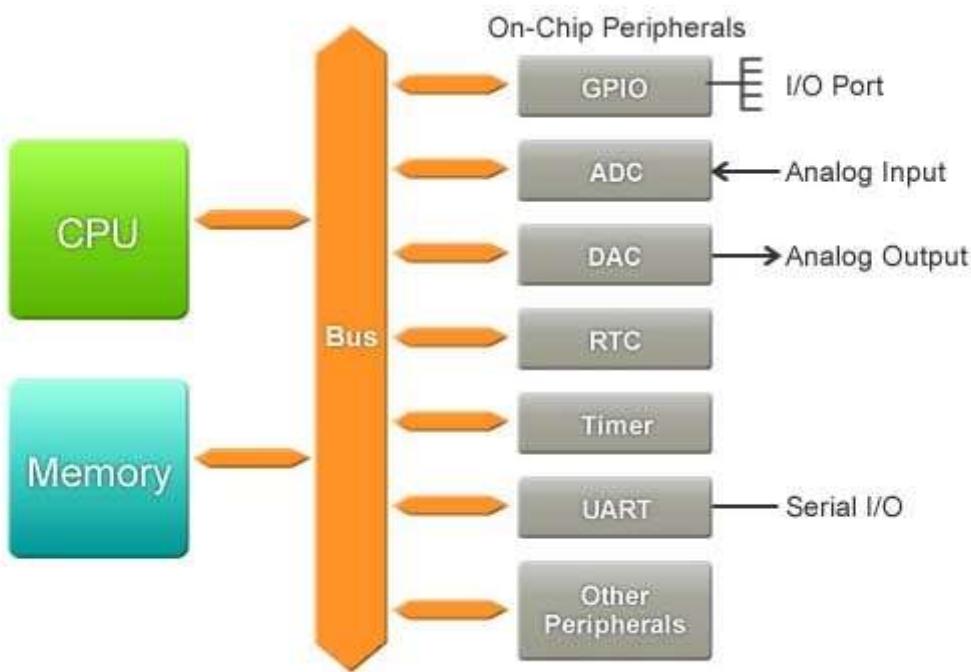
khiển, không bị mất khi tắt điện hoặc reset.



## Timer/counter

Một vi điều khiển có thể có nhiều bộ đếm thời gian và bộ đếm. Bộ đếm thời gian và bộ đếm có chức năng đếm thời gian tạo ra các sự kiện để vi điều khiển hoạt động đúng thời điểm.

## Các ngoại vi của vi điều khiển



*Ngoại vi của vi điều khiển*

## I/O Ports – Input/output

Có thể coi I/O Port là tay chân của vi điều khiển, chúng giúp cho vi điều khiển tương tác với các thành phần khác ngoài môi trường.

Cổng đầu vào / đầu ra được sử dụng chủ yếu để điều khiển hoặc giao tiếp các thiết bị như màn hình LCD, đèn LED, máy in, ... cho vi điều khiển.

## Các chuẩn giao tiếp

Giống như miệng và tai vậy. Vi điều khiển sẽ sử dụng các chuẩn giao tiếp khác nhau để liên lạc với nhau hoặc liên lạc với các phẳng tử khác trên mạch. Có thể kể đến như I2C, SPI, UART, USB, ....

## Bộ chuyển đổi analog sang digital (ADC)

Bộ chuyển đổi ADC được sử dụng để chuyển đổi tín hiệu analog sang dạng digital. Tín hiệu đầu vào trong bộ chuyển đổi này phải ở dạng analog (ví dụ: đầu ra cảm biến) và đầu ra từ thiết bị này ở dạng digital. Đầu ra digital có thể được sử dụng cho các ứng dụng kỹ thuật số (ví dụ: các thiết bị đo lường).

## Bộ chuyển đổi Digital sang Analog (DAC)

Hoạt động của DAC là đảo ngược của ADC. DAC chuyển đổi tín hiệu digital thành định dạng analog. Nó thường được sử dụng để điều khiển các thiết bị analog như động cơ DC, các ổ đĩa...

## Interrupt control hay quản lý sự kiện

Ngoài việc thực thi chương trình, vi điều khiển còn phải tương tác với các tác nhân bên trong và bên ngoài. Các tác nhân này sẽ tạo ra các sự kiện gọi là Ngắt, để quản lý nó cần có một khối quản lý ngắt ( Interrupt control)

## Special functioning block

Một số vi điều khiển chỉ được sử dụng cho một số ứng dụng đặc biệt (ví dụ: hệ thống không gian và rô bốt) các bộ điều khiển này có chứa các cổng bổ sung để thực hiện các hoạt động đặc biệt đó. Đây được coi là khối chức năng đặc biệt.

## Tiếp cận với vi điều khiển như thế nào?

Vậy để bắt đầu lập trình vi điều khiển chúng ta cần làm những gì? Cùng tìm hiểu nhé!

## Chọn dòng vi điều khiển nào?

Khi đặt câu hỏi này, chúng ta nghĩ ngay đến tính năng, số chân, và kích thước cần thiết của vi điều khiển. Và chúng ta phải lựa chọn được con vi điều khiển chúng ta cần dùng, tất nhiên kèm theo ngay sau đó là chúng ta có thể mua được nó nữa.

Tùy theo ứng dụng, giá cả, chức năng, độ ổn định chúng ta cần chọn cho mình một hoặc 2 loại để bắt đầu.

- Nếu bạn muốn học sâu về vi điều khiển mình khuyên các bạn nên học từ những con đơn giản như 8051, lập trình sử dụng thanh ghi của nó. Bạn sẽ hiểu sâu về vi điều khiển, sau đó thì có thể chuyển qua dòng khác một cách rất đơn giản.

- Nếu bạn muốn sử dụng nó để làm sản phẩm, các bạn có thể chọn STM32, STM8,... Các dòng vi điều khiển này có bộ thư viện và công cụ giúp chúng ta làm sản phẩm 1 cách nhanh chóng.

Tham khảo: [Học lập trình STM32 từ A tới Z](#)

- Nếu bạn muốn làm các ứng dụng IOT các bạn nên sử dụng các chip có hỗ trợ các chuẩn truyền thông không dây (wifi, ble, zigbee...) như ESP32, ESP8266, NRF52832....

Tham khảo: [Học lập trình ESP32 từ A tới Z](#)

- Còn nếu bạn chỉ muốn DIY các sản phẩm đơn giản hoặc dùng để làm quen với lập trình nhúng, các bạn có thể sử dụng [Arduino](#) hay các dòng có hỗ trợ thư viện Arduino.

## Lập trình cho dòng vi điều khiển đó

Nếu vi điều khiển chỉ là thân xác, thì việc lập trình chính là các bạn đang tạo ra linh hồn cho nó.

Để lập trình vi điều khiển thì bắt buộc các bạn phải học ngôn ngữ C, vì ngôn ngữ C có thể can thiệp tới tầng thấp nhất của phần cứng, điều mà các ngôn ngữ khác không làm được.

Tham khảo: [Lập trình C từ A tới Z](#)

Bản chất của lập trình vi điều khiển chỉ là tạo ra các hành động cụ thể cho nó. Như việc con người chúng ta tương tác với thế giới xung quanh như thế nào vậy. Mọi thao tác đó được lập trình viên viết ra bằng ngôn ngữ lập trình C hoặc ngôn ngữ khác.

Sau đó ngôn ngữ đó được thông dịch lại cho vi điều khiển hiểu, quá trình đó gọi là biên dịch. Thường thì sẽ tạo ra file .hex hoặc .bin

Muốn làm được điều này, các bạn cần có một trình biên dịch, hoặc môi trường lập trình tích hợp (IDE). Có thể kể đến như [KeilC](#), [Arduino](#), [VScode](#), ....

## Nạp chương trình

Bạn viết chương trình trên máy tính, bạn đã dịch ra được file thực thi .hex, để vi điều khiển có thể hiểu được bạn muốn làm gì. Vậy làm sao để đưa nội dung đó vào cho vi điều khiển?

Các bạn cần có một mạch nạp và một chương trình nạp phù hợp với mạch nạp đó. Công việc nạp được cụ thể hóa bằng việc cắm mạch nạp vào máy tính, bật chương trình nạp, load file .HEX vào chương trình nạp, lựa chọn vi điều khiển cần nạp, cài đặt các thông số nạp, và nạp vào vi điều khiển đó

Các mạch nạp có thể kể đến như: ST Link, JTAG, ISP, ....

## Debug chương trình

Cuối cùng là công đoạn gian nan mà mỗi lập trình viên đều phải làm, đó là Debug. Hay nói cách khác là sửa những lỗi lập trình khiến code của bạn không hoạt động đúng.

Một lập trình viên giỏi không phải là người viết code nhanh, viết đc nhiều code. Mà là người có thể fix đc hết Bug hoặc nhiều Bug nhất có thể.

Tham khảo: [Các công cụ Debug trên Keil C](#)

## Ưu và nhược điểm của vi điều khiển

### Ưu điểm của vi điều khiển

- Những ưu điểm chính của vi điều khiển là:
- Vi điều khiển hoạt động như một máy vi tính không có bất kỳ bộ phận kỹ thuật số nào.
- Tích hợp cao hơn bên trong vi điều khiển làm giảm chi phí và kích thước của hệ thống.
- Việc sử dụng vi điều khiển rất đơn giản, dễ khắc phục sự cố và bảo trì hệ thống.

- Hầu hết các chân được lập trình bởi người dùng để thực hiện các chức năng khác nhau.
- Dễ dàng kết nối thêm các cổng RAM, ROM, I/O.
- Cần ít thời gian để thực hiện các hoạt động.

## Nhược điểm của vi điều khiển

- Vi điều khiển có kiến trúc phức tạp hơn so với vi xử lý.
- Chỉ thực hiện đồng thời một số lệnh thực thi giới hạn.
- Chủ yếu được sử dụng trong các thiết bị vi mô.
- Không thể trực tiếp giao tiếp các thiết bị công suất cao.

## Ứng dụng của vi điều khiển

Vi điều khiển hiện hữu trên rất nhiều mặt của cuộc sống.

Bạn có thể tìm thấy vi điều khiển trong tất cả các loại thiết bị điện tử hiện nay. Bất kỳ thiết bị nào liên quan đến đo lường, lưu trữ, điều khiển, tính toán hoặc hiển thị thông tin đều phải có chip vi điều khiển bên trong.

Ứng dụng lớn nhất của vi điều khiển là trong ngành công nghiệp ô tô (vi điều khiển được sử dụng rộng rãi để kiểm soát động cơ và điều khiển công suất trong ô tô).

Bạn cũng có thể tìm thấy vi điều khiển bên trong bàn phím, chuột, modem, máy in và các thiết bị ngoại vi khác. Trong thiết bị thử nghiệm, vi điều khiển giúp bạn dễ dàng thêm các tính năng như khả năng lưu trữ số đo, tạo và lưu trữ các thói quen của người dùng và hiển thị thông báo cũng như dạng sóng.

Sản phẩm tiêu dùng sử dụng bộ vi điều khiển bao gồm máy quay kỹ thuật số, đầu phát quang, màn hình LCD / LED...

Hay đến với thời đại 4.0 các bạn sẽ thấy vi điều khiển trong các thiết bị IoT, giúp con người kết nối mọi máy móc từ xa thông qua Internet



## Kết

Hiểu được cấu trúc của vi điều khiển là một phần tất yếu khi học **Lập trình nhúng**. Nếu bạn vẫn đang mông lung khi gặp các khái niệm mình đã nêu trên thì nên tìm hiểu kĩ càng lại từ đầu. Bởi nếu không hiểu chúng ta đang làm việc với cái gì, thì rất khó để làm nó chạy một cách chính xác, rất khó tìm ra lỗi, nguyên nhân khiến code của bạn không chạy.

Cám ơn bạn đã đón đọc, cùng vào hội **Anh Em Nghiện Lập Trình** để cùng trao đổi nhé

5/5 - (2 bình chọn)

### Related Posts:

1. **CPU là gì? Cấu tạo và nguyên lý hoạt động của CPU**





## KHUÊ NGUYỄN

Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

## Trả lời

Email của bạn sẽ không được hiển thị công khai. Các trường bắt buộc được đánh dấu \*

Bình luận \*

Tên \*

Email \*

Trang web

PHẢN HỒI

Fanpage

 Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển  
2.754 lượt thích

**Đã thích** **Chia sẻ**

**Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển**  
khoảng một tháng trước

Lý do thời gian gần đây mình không viết bài và làm thêm gì cả là đây 😊  
Chính thức ra mắt sản phẩm định vị thông minh vTag.

Đây là một sản phẩm định vị đa năng với 3 công nghệ định vị WIFI, GPS, LBS kết hợp với sóng NB-IOT dành riêng cho các sản phẩm IOT.

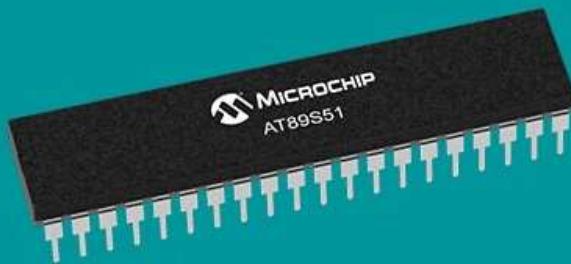
Chỉ với 990.000đ chúng ta đã có thể có sản phẩm để:

- Định vị trẻ em, con cái... Xem thêm



## Bài viết khác

### Lập trình 8051 - AT89S52



**Bài 1: Tổng quan về 8051 và chip AT89S51 - 52**



**Khuê Nguyễn Creator**



## Tổng quan về 8051

8051 là một dòng chip nhập môn cho lập trình viên nhúng, chúng được sử...

[ĐỌC THÊM](#)

## Lập trình STM32 và CubeMX



Khuê Nguyễn Creator

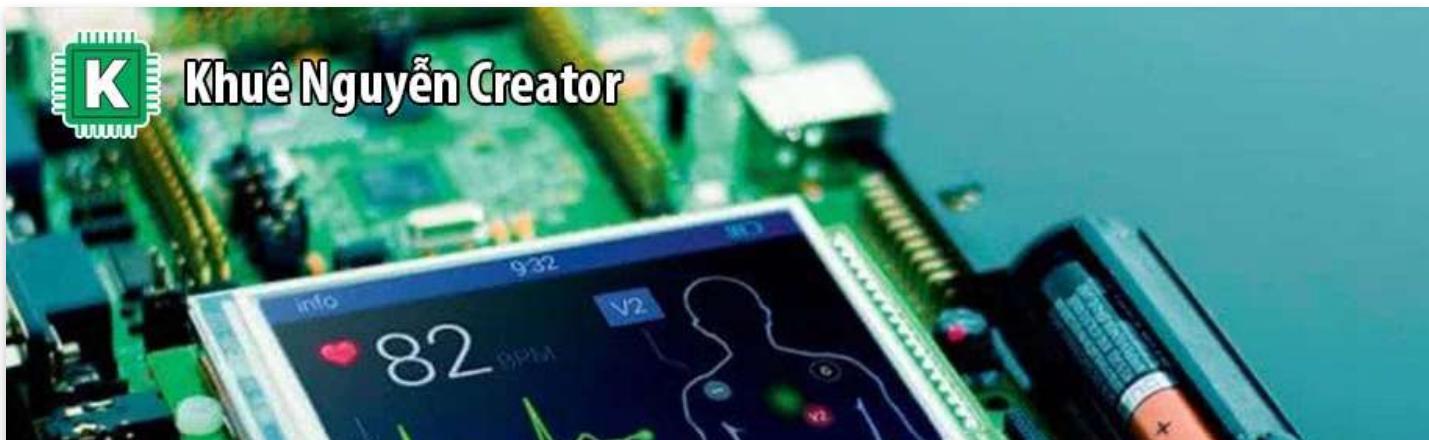


## Lập trình STM32 HID Host giao tiếp với chuột và bàn phím

Lập trình STM32 USB HID Host giao tiếp với chuột và bàn phím máy tính

Trong bài này chúng ta sẽ cùng học STM32 HID Host, biến STM32 giống như...

[ĐỌC THÊM](#)





## Lộ trình học lập trình nhúng từ A tới Z

Lập trình nhúng là một ngành có cơ hội nhưng cũng đòi hỏi nhiều kiến...

3 COMMENTS

[ĐỌC THÊM](#)

### Lập trình STM32 và CubeMX





### Khuê Nguyễn Creator



### Lập trình STM32F407 SDIO đọc dữ liệu thẻ nhớ

#### Lập trình STM32 SDIO đọc ghi dữ liệu vào thẻ nhớ SD card

Trong bài này chúng ta cùng học cách lập trình STM32 SDIO, một chuẩn giao...

[ĐỌC THÊM](#)

[Lập trình STM32 và CubeMX](#)



Khuê Nguyễn Creator

## Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



# Lập trình STM32F407 DAC chuyển đổi số sang tương tự

**Lập trình STM32 DAC tạo sóng hình Sin trên KIT STM32F407 Discovery**

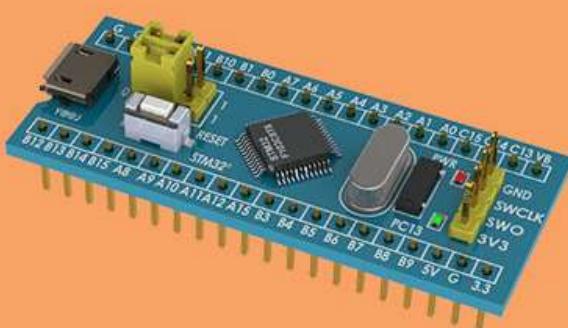
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[ĐỌC THÊM](#)

## Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



# Sử dụng hàm printf để in Log khi Debug trên STM32

**Hướng dẫn sử dụng printf với STM32 Uart để in Log trên Keil C**

Trong bài này chúng ta sẽ học cách retarget hàm printf của thư viện stdio...

3 COMMENTS

[ĐỌC THÊM](#)

## ESP32 và Platform IO



Khuê Nguyễn Creator



## Bài 9 WIFI: Lập trình ESP32 OTA nạp firmware trên Internet

Lập trình ESP32 FOTA nạp firmware qua mạng Internet với OTA Drive

Trong bài này chúng ta sẽ học cách sử dụng ESP32 FOTA (Firmware Over The...

4 COMMENTS

[ĐỌC THÊM](#)

## Lập trình Nuvoton



Khuê Nguyễn Creator

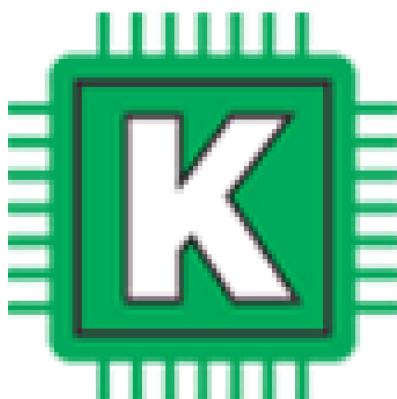


# Cài đặt SDC Complier và Code:Blocks IDE

Hướng dẫn cài đặt SDCC và Code::Blocks lập trình Nuvoton

Ở bài này chúng ta sẽ cài đặt các công cụ cần thiết cho việc...

[ĐỌC THÊM](#)



**KHUÊ NGUYỄN CREATOR**  
**Chia sẻ đam mê**

Blog này làm ra để lưu trữ tất cả những kiến thức, những câu chuyện của mình. Đôi khi là những ý tưởng nhất thời, đôi khi là các dự án tự mình làm. Chia sẻ cho người khác cũng là niềm vui của mình, kiến thức mỗi người là khác nhau, không hẳn quá cao siêu nhưng sẽ có lúc hữu dụng.

## Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

## Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn