**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA ĐIỆN TỬ**

-----------------------------------



**ĐỒ ÁN TỐT NGHIỆP**

***Đề tài****: “ Nghiên cứu và ứng dụng vi điều khiển ARM trong thiết kế mạnh điều khiển học lênh từ xa các thiết bị trong gia đình”*

Người hướng dẫn: TS. Hà Thu Lan  
Sinh viên thực hiện: BÙI QUANG TÙNG  
Lớp: D12XLTH  
Hệ: Đại học chính quy

# Lời cảm ơn

Để hoàn thành đồ án tốt nghiệp này em xin gửi lời cảm ơn chân thành tới cô Hà Thu Lan đã tận tình hướng dẫn em trong suốt quá trình thực hiện đồ án tốt nghiệp này.  
  
Em cũng xin chân thành cảm ơn tới quý thầy, cô Học viện Công nghệ Bưu chính Viễn thông, đặc biệt là các thầy, cô trong khoa Kỹ Thuật Điện Tử I đã nhiệt tình giúp đỡ, truyền đạt kiến thức trong suốt quá trình học tập của em tại Học viện. Vốn kiến thức được tiếp thu trong quá trình học tập không chỉ là nền tảng cho quá trình thực hiện đồ án tốt nghiệp mà còn là hành trang quý báu cho sự nghiệp của em sau này.  
  
Em cũng xin cảm ơn sự ủng hộ và giúp đỡ nhiệt tình của gia đình, bạn bè, những người thân đã động viên, giúp đỡ em trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp này.  
  
Mặc dù đã cố gắng hết sức, song đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và chỉ bảo tận tình của quý thầy cô và các bạn để em có thể hoàn thành tốt hơn đồ án tốt nghiệp này.  
  
Cuối cùng em xin kính chúc quý Thầy, Cô, gia đình và bạn bè dồi dào sức khỏe và thành công trong sự nghiệp.

# Mục lục

Lời cảm ơn 2

Mục lục 3

Danh mục các ký hiệu và chữ viết tắt 6

Danh mục hình ảnh 7

Danh mục bảng 8

Lời mở đầu 9

Chương 1: Giới thiệu bộ xử lý ARM Cortex M3 10

1. Giới thiệu 10

1.1. Hiệu suất cao 10

1.2. Dễ sử dụng, phát triển ứng dụng nhanh chóng, hiệu quả 11

1.3. Giảm chi phí phát triển và năng lương tiêu thụ 11

1.4. Tích hợp khả năng dò lỗi, theo vết trong lập trình 12

2. Kiến trúc và tính năng xử lý 12

2.1. Lõi Cortex M3 12

2.2. Kiến trúc tập lệnh Thumb-2 16

2.3. Bộ điều khiển vector ngắt lồng nhau (NVIC) 17

2.4. Đơn vị bảo vệ bộ nhớ 19

2.5. Gỡ lỗi ( debug) và theo vết (trace) 20

2.6. Ma trận bus và các giao diện liên kết 22

Chương 2: NEC Infrared Transmission Protocol 23

1. NEC protocol 23

2. Đặc điểm 23

3. Điều chế 23

4. Giao thức 24

5. Giao thức mở rộng NEC 25

Chương 3: Hồng ngoại và ứng dụng 27

1. Phân loại 27

2. Nguồn phát tia hồng ngoại 29

3. Tính chất 29

4. Ứng dụng 30

4.1 Đo nhiệt độ 30

4.2 Phát nhiệt 30

4.3 Kỹ thuật hồng ngoại trong quân sự 30

4.4 Truyền thông 30

4.5 Thiết bị nhìn đêm 31

4.6 Nghiên cứu thiên văn 32

4.7 Bảo mật tiền và tài liệu quý 32

4.8 Điện tử điều khiển 32

4.8.1 Tự động bật tắt thiết bị 32

4.8.2 Phụ kiện thiết bị vi tính 33

4.8.3 Các điều khiên xa (remote control) 33

Chương 4: Thiết kế mạch điều khiển từ xa bằng hồng ngoại 34

1. Giới thiệu 34

1.1 Vi điều khiển STM32F103 34

1.1.1 Overview 35

1.1.2 Họ vi điều khiển của STM32F103xx 36

1.2 Module sạc pin có bảo vệ dùng TP4056 46

1.2.1 Tính năng 46

1.2.2 Thông số kỹ thuật 46

1.3 Module tăng áp 47

1.3.1 Tính năng 47

1.3.2 Thông số kỹ thuật 48

1.4 Màn hình LCD01006A 48

1.4.1 Tính năng 48

1.4.2 Thông số kỹ thuật 49

2 Sơ đồ khối và nguyên lý 49

2.1 Sơ đồ khối 49

2.2 Nguyên lý hoạt động 50

2.2.1 Chế độ học lệnh 50

2.2.2 Chế độ điều khiển 50

3 Thiết kế mạch nguyên lý và mạch in 50

3.1 Mạch nguyên lý 50

3.2 Mạch in 52

4 Thuật toán xử lý và lập trình 53

4.1 Thuật toán 53

4.1.1 Lưu đồ thuật toán 53

4.1.2 Giải thích thuật toán 54

4.2 Lập trình 55

4.2.1 Hàm ghi dữ liệu 55

4.2.2 Hàm đưa dữ liệu lên màn hình 55

4.2.3 Hàm GetByte 56

4.2.4 NOT\_BIT\_BYTE 56

4.2.5 Hàm cấu hình ngắt 57

4.2.6 Hàm cấu hình IO 57

4.2.7 Hàm cấu hình UART 58

5 Kết quả 60

Kết luận 62

Tham khảo 63

# Danh mục các ký hiệu và chữ viết tắt

|  |  |
| --- | --- |
| Soc (System-on-chip) | Hệ thống tích hợp trên chip |
| ICE (In-circuit Emulator) | Bộ mô phỏng mạch điện |
| SWD (Serial Wire Debug) | Cổng debug serial |
| RAM (Random access memory) | Bộ nhớ truy cập ngẫu nhiên |
| NVIC (Nested Vectored Interrupt Controller) | Điều khiển vector ngắt lồng nhau |
| MPU(Main Processing Unit MicroProcessor) | Bộ xử lý đơn vị |
| DAP (Debug Access Port) | Cổng truy cập debug |
| SWJ-D (Serial Wire JTAG Debug) | Cổng debug qua JTAG |
| ADC ( Analog to Digital converter) | Chuyển đổi tương tự sang tín hiệu số |
| PWM (Pulse Width Modulation ) | Bộ băm xung ( điều rộng xung ) |
| USART (Universal Synchronous/Asynchronous Receiver/Transmitter) | Bộ truyền nhận nối tiếp đồng bộ / không đồng bộ |
| AHB (Advanced High Speed Buses) | Bus kết nối hệ thống tốc độ cao |
| CPU (Central Processing Unit) | Bộ điều khiển trung tâm |
|  |  |

# Danh mục hình ảnh

|  |  |
| --- | --- |
| Hình ảnh | Trang |
| Hình 1.1 Bộ vi xử lý Cortex M3 | 13 |
| Hình 1.2 The Memory Map | 15 |
| Hình 1.3 Comparison of traditional bit manipulation with Cortex-M3- banding | 16 |
| Hình 1.4 Relative Dhrystone performance and code size for ARM, Thumb, Thumb-2 | 17 |
| Hình 1.5 Tail chaining trong NVIC | 19 |
| Hình 1.6 Hệ thống theo vết Cortex-M3 | 21 |
| Hình 2.1 Chuẩn giao thức NEC sử dụng mã hóa độ rộng xung của các bit | 23 |
| Hình 2.2 Dãy xung của giao thức NEC | 24 |
| Hình 2.3 LBS | 25 |
| Hình 2.4 Repeat code | 25 |
| Hình 2.5 Address-16bit | 26 |
| Hình 4.1 Vi điều khiển STM32F103 | 34 |
| Hình 4.2 STM32F103xx pinout | 36 |
| Hình 4.3 STM32F103 Performance line Block Diagram | 38 |
| Hình 4.4 System Architecture | 39 |
| Hình 4.5 STM32F103 Memory Map | 41 |
| Hình 4.6 Clock tree for microcontroller STM32F103xx | 43 |
| Hình 4.7 Pin Configuration of the STM32F103 | 45 |
| Hình 4.8 Module sạc pin dùng TP4056 | 46 |
| Hình 4.9 Module tăng áp | 47 |
| Hình 4.10 LCD01006A | 48 |
| Hình 4.11 Sơ đồ khối của mạch điều khiển từ xa bằng hồng ngoại | 49 |
| Hình 4.12 Khối điều khiển | 50 |
| Hình 4.13 Khối thu phát tín hiệu và mạch nguồn | 51 |
| Hình 4.14 Mạch nguyên lý | 52 |
| Hình 4.15 Mạch in (bottom) | 52 |
| Hình 4.16 Mạch in (top) | 53 |

# Danh mục bảng

|  |  |
| --- | --- |
| Bảng | Trang |
| Bảng 3.1 Bảng phân chia các bức xạ sóng điện từ/ ánh sáng | 27 |
| Bảng 3.2 Bảng phân loại tia hồng ngoại thông dụng theo phân loại Mỹ | 29 |
| Bảng 3.3 Bảng băng tần viễn thông IR | 31 |
| Bảng 3.4 Phân loại các nhóm theo chức năng STM32F103xx | 37 |

# Lời mở đầu

Trong nhiều năm trước, các dòng vi điều khiển 8051 được sinh viên dùng nhiều với tính năng đơn giản, dễ sử dụng: AVR sử dụng nhiều trong các cuộc thi thiết kế điện tử với tốc độ xử lý cao, ổn định; PIC với ưu thế tốc độ và sự ổn định với chi phí thấp. Nhưng các lĩnh vực có yêu cầu về tốc độ cao hơn, độ ổn định, tiêu thụ năng lương chặt chẽ hơn đòi hỏi vi điều khiển tiên tiến hơn. Và họ vi điều khiển ARM ra đời với nhiều thế hệ, phiên bản khác nhau tùy theo mục đích sử dụng và ưu tiên khác nhau.

*Lý do chọn đề tài*: Với nhiều tính năng vượt trội của dòng ARM này, cũng là xu hướng lựa chọn trong các hệ thống nhúng, dưới sự giúp đỡ của cô Hà Thu Lan, em thực hiện nghiêu cứu ứng dụng của chip ARM Cortex M3 STM32F103 trong thiết kế mạch học lệnh và điều khiển các thiết bị từ xa qua hồng ngoại.

*Mục đích nghiên cứu*: Việc sử dụng quá nhiều điều khiển cho các thiết bị điện trong gia đình gây ra những phiền toái khi sử dụng. Việc có 1 mạch có khả năng điều khiển tất cả các thiết bị điện trong phòng sẽ loại bỏ sự phiền toái đồng thời giảm nguy cơ xảy ra hư hỏng với thiết bị remote này.

*Tình hình nghiên cứu*: Ứng dụng vi điều khiển STM32F103 để tạo ra 1 mạch có khả năng học lệnh và từ đó điều khiển các thiết bị trong phòng qua sóng hồng ngoại.

*Phạm vi đề tài*: Mạch học lệnh và điều khiển các thiết bị dung chuẩn giao tiếp phổ biến NEC

# Chương 1: Giới thiệu bộ xử lý ARM Cortex M3

1. Giới thiệu

Giải pháp Soc (System-on-chip) dựa trên bộ vi xử lý nhúng ARM được ứng dụng vào rất nhiều thị trường khác nhau bao gồm các ứng dụng doanh nghiệp, các hệ thống ô tô, mạng gia đình và công nghệ mạng không dây... Dòng vi xử lý ARM Cortex dựa trên một kiến trúc chuẩn đủ để đáp ứng hầu hết các yêu cầu về hiệu năng làm việc trong tất cả các lĩnh vực trên. Dòng ARM Cortex bao gồm ba cấu hình khác nhau của kiến trúc ARMv7: cấu hình A cho các ứng dụng tinh vi, yêu cầu cao chạy trên các hệ điều hành mở và phức tạp như Linux, Android…; cấu hình R dành cho các hệ thống thời gian thực và cấu hình M được tối ưu cho các ứng dụng vi điều khiển, cần tiết kiệm chi phí. Bộ vi xử lý Cortex-M3 là bộ vi xử lý ARM đầu tiên dựa trên kiến trúc ARMv7-M và được thiết kế đặc biệt để đạt được hiệu suất cao trong các ứng dụng nhúng cần tiết kiệm năng lượng và chi phí, chẳng hạn như các vi điều khiển, hệ thống cơ ô tô, hệ thống kiểm soát công nghiệp và hệ thống mạng không dây. Thêm vào đó là việc lập trình được đơn giản hóa đáng kể giúp kiến trúc ARM trở thành một lựa chọn tốt cho ngay cả những ứng dụng đơn giản nhất.

* 1. **Hiệu suất cao**

Để đạt được hiệu suất cao hơn, bộ vi xử lý có thể làm việc nhiều hơn hoặc làm việc thông minh hơn. Đẩy tần số hoạt động cao hơn có thể làm tăng hiệu suất nhưng cũng đi kèm với việc tiêu thụ năng lượng nhiều hơn và việc thiết kế cũng phức tạp hơn. Nói cách khác, cùng thực hiện những tác vụ đó nhưng bằng cách nâng cao hiệu quả tính toán trong khi vẫn hoạt động ở tần số thấp sẽ dẫn đến sự đơn giản hóa trong việc thiết kế và ít tốn năng lượng hơn. Trung tâm của bộ vi xử lý Cortex-M3 là một lõi có cấu trúc đường ống tiên tiến 3 tầng, dựa trên kiến trúc Harvard, kết hợp nhiều tính năng mới mạnh mẽ như suy đoán việc rẽ nhánh, phép nhân được thực thi trong một chu kỳ và phép chia được thực hiện bằng phần cứng tạo nên một hiệu năng vượt trội (điểm Dhrystone là 1,25 DMIPS/MHz). Bộ vi xử lý Cortex-M3 hỗ trợ kiến trúc tập lệnh Thumb-2, giúp nó hoạt động hiệu quả hơn 70% cho mỗi MHz so với một bộ vi xử lý ARM7TDMI-S thực thi với tập lệnh Thumb, và hiệu quả hơn 35% so với bộ xử lý ARM7TDMI-S thực thi với tập lệnh ARM.

* 1. **Dễ sử dụng, phát triển ứng dụng nhanh chóng, hiệu quả**

Tiêu chí quan trọng trong việc lựa chọn bộ vi xử lý là giảm thời gian và chi phí phát triển, đặc biệt là khả năng phát triển ứng dụng phải thật nhanh chóng và đơn giản. Bộ vi xử lý Cortex-M3 được thiết kế để đáp ứng mục tiêu trên. Người lập trình không cần phải viết bất kì mã hợp ngữ nào (assembler code) hoặc cần phải có kiến thức sâu về kiến trúc để tạo ra một ứng dụng đơn giản. Bộ vi xử lý có mô hình lập trình dựa trên ngăn xếp đã được đơn giản hoá để tương thích với kiến trúc ARM truyền thống nhưng tương tự với hệ thống đã được triển khai trên kiến trúc 8 và 16-bit, giúp việc chuyển tiếp đến kiến trúc 32-bit dễ dàng hơn. Ngoài ra một mô hình ngắt dựa trên phần cứng sẽ giúp việc viết các chương trình xử lý ngắt trở nên đơn giản hơn bao giờ hết, chương trình khởi động có thể được viết trực tiếp bằng ngôn ngữ C mà không cần bất kì một lệnh assembly nào so với kiến trúc ARM truyền thống. Các tính năng chính mới trong tập lệnh Thumb-2 bao gồm việc thực hiện mã lệnh C một cách tự nhiên hơn, thao tác trực tiếp trên các bit, phép chia phần cứng và lệnh If/Then. Hơn nữa, nhìn từ góc độ phát triển ứng dụng, Thumb-2 tăng tốc độ phát triển, đơn giản hóa việc bảo trì, hỗ trợ các đối tượng biên dịch thông qua tối ưu hóa tự động cho cả hiệu suất và mật độ mã mà không cần quan tâm đến việc mã được biên dịch cho chế độ ARM hoặc Thumb. Kết quả là lập trình viên có thể để mã nguồn của họ trong ngôn ngữ C mà không cần tạo ra các thư viện đối tượng biên dịch sẵn, có nghĩa là khả năng tái sử dụng mã nguồn lớn hơn nhiều.

* 1. **Giảm chi phí phát triển và năng lương tiêu thụ**

Chi phí luôn là rào cản lớn nhất cho sự lựa chọn một bộ vi xử lý hiệu suất cao. Bộ vi xử lý được thiết kế trên một diện tích nhỏ sẽ giảm chi phí đáng kể. Bộ vi xử lý Cortex-M3 thực hiện điều này bằng cách cài đặt các lõi ARM nhỏ nhất từ trước đến nay, chỉ với 33.000 cổng (cổng có thể là NAND hoặc NOR… tuỳ vào công nghệ sản xuất) trong lõi trung tâm (0.18um G) và bằng cách kết hợp hiệu quả, chặt chẽ các thành phần trong hệ thống vi xử lý. Bộ nhớ được tối giản bằng cách cài đặt bộ nhớ không thẳng hàng (unaligned), thao tác bit dễ dàng với kĩ thuật bit banding. Tập lệnh Thumb-2 tiết kiệm bộ nhớ hơn 25% so với tập lệnh ARM.  
Để đáp ứng nhu cầu ngày càng tăng trong việc tiết kiệm năng lượng ở các ứng dụng mạng không dây…, bộ vi xử lý Cortex-M3 hỗ trợ mở rộng xung nhịp cho các cổng (có thể ngừng cung cấp xung nhịp cho các cổng để tiết kiệm năng lượng) và tích hợp chế độ ngủ. Kết quả là bộ vi xử lý chỉ tiêu thụ 4.5mW điện năng và chiếm diện tích 0.3 mm2 (silicon footprint) khi triển khai ở tần số 50MHz trên quá trình công nghệ TSMC 0.13G, sử dụng tế bào tiêu chuẩn ARM Metro.

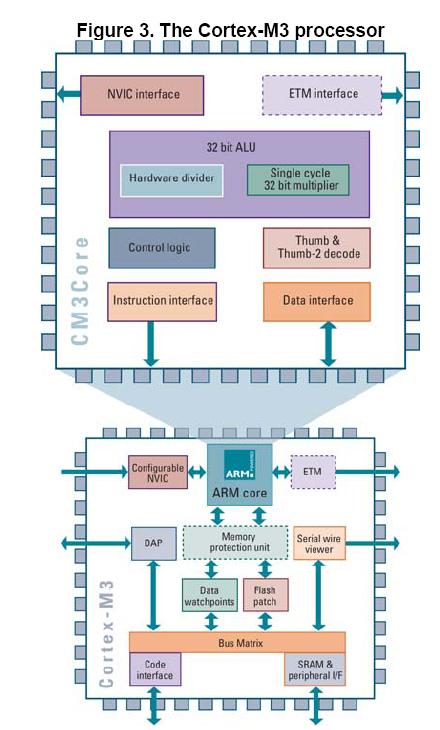
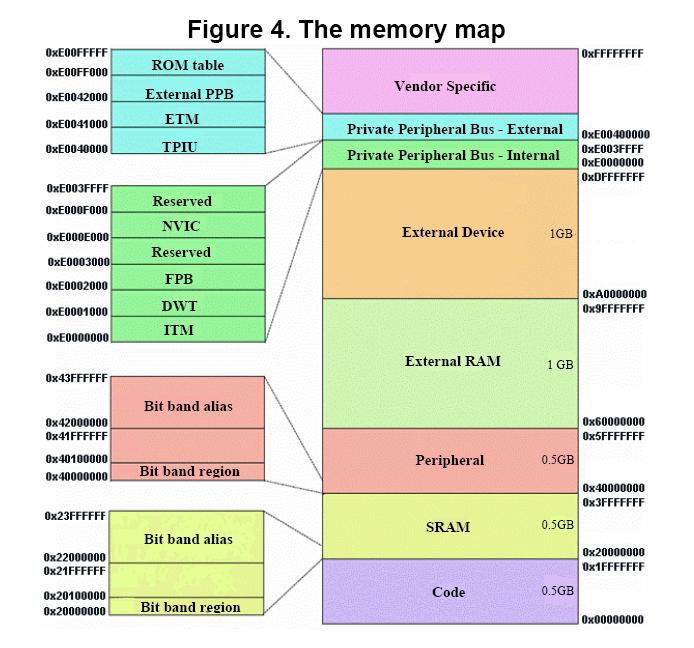
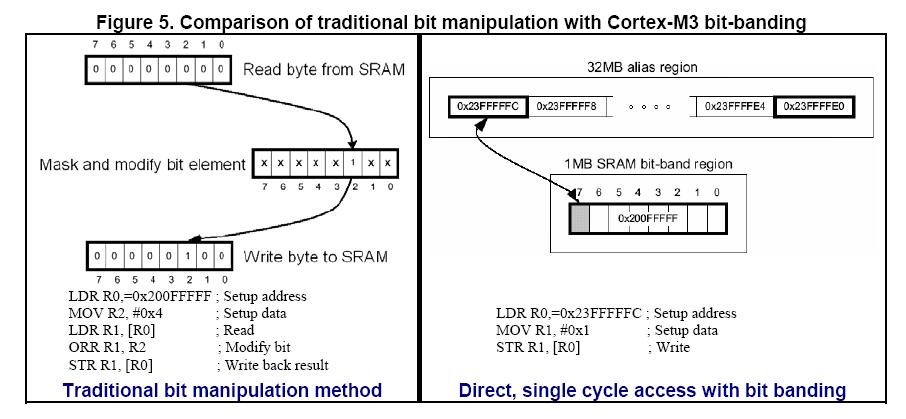
* 1. **Tích hợp khả năng dò lỗi, theo vết trong lập trình**

Hệ thống nhúng thường không có giao diện người dùng đồ họa (GUI) làm cho việc gỡ lỗi chương trình trở thành một thách thức thật sự đối các lập trình viên. Ban đầu, bộ ICE (In-circuit Emulator) đã được sử dụng để tạo một cửa sổ theo dõi hệ thống thông qua một giao diện quen thuộc như trên PC. Tuy nhiên khi hệ thống ngày càng nhỏ và phức tạp hơn, phương pháp này không còn khả thi nữa. Công nghệ gỡ lỗi của bộ vi xử lý Cortex-M3 được cài đặt trong chính phần cứng của nó (kết hợp với một vài thành phần khác) giúp gỡ lỗi nhanh hơn với các tính năng trace & profiling, breakpoints, watchpoints và bản vá lỗi giúp rút ngắn thời gian phát triển ứng dụng. Ngoài ra, bộ vi xử  lý còn cung cấp một mức nhìn cao hơn vào hệ thống thông qua cổng JTAG truyền thống hoặc cổng SWD (Serial Wire Debug) chỉ sử dụng 2 đường tín hiệu, thích hợp cho các thiết bị có kiểu đóng  gói nhỏ gọn.

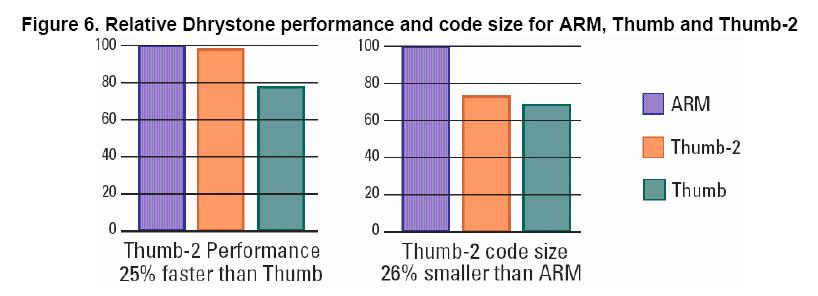
1. **Kiến trúc và tính năng xử lý**

Bộ vi xử lý Cortex-M3 dựa trên kiến trúc ARMv7-M có cấu trúc thứ bậc. Nó tích hợp lõi xử lý trung tâm, gọi là CM3Core, với các thiết bị ngoại vi hệ thống tiên tiến để tạo ra các khả năng như kiểm soát ngắt, bảo vệ bộ nhớ, gỡ lỗi và theo vết hệ thống.  
Các thiết bị ngoại vi có thể được cấu hình một cách thích hợp, cho phép bộ vi xử lý Cortex-M3 đáp ứng được rất nhiều ứng dụng và yêu cầu khắt khe của hệ thống. Lõi của bộ vi xử lý Cortex-M3 và các thành phần tích hợp đã được thiết kế đặc biệt để đáp ứng yêu cầu bộ nhớ tối thiểu, năng lượng tiêu thụ thấp và thiết kế nhỏ gọn.

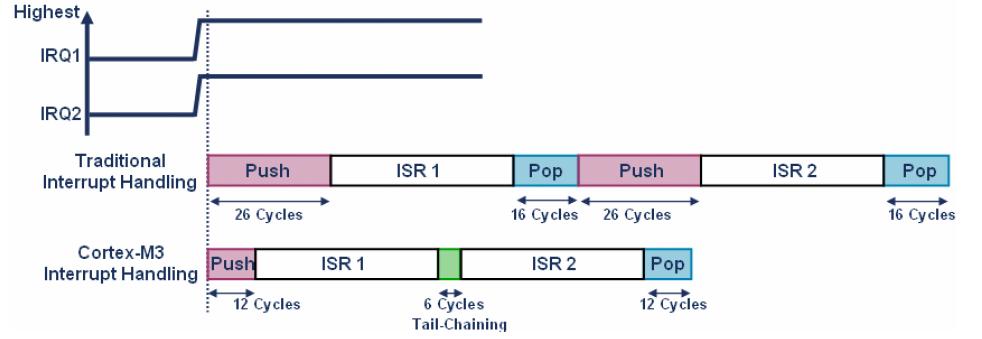
* 1. **Lõi Cortex M3**

Lõi trung tâm Cortex-M3 dựa trên kiến trúc Harvard, được đặc trưng bằng sự tách biệt giữa vùng nhớ chứa dữ liệu và chương trình do đó có các bus riêng để truy cập. Đặc tính này khác với dòng ARM7 dựa trên kiến trúc Von Neumann sử dụng chung vùng nhớ để chứa dữ liệu và chương trình, do đó dùng chung bus cho việc truy xuất. Vì có thể đọc cùng lúc lệnh và dữ liệu từ bộ nhớ, bộ vi xử lý Cortex-M3 có thể thực hiện nhiều hoạt động song song, tăng tốc thực thi ứng dụng.  
****  
Hình 1.1: Bộ vi xử lý Cortex M3   
  
Lõi Cortex có cấu trúc đường ống gồm 3 tầng: **Instruction Fetch**, **Instruction Decode** và **Instruction Execute**. Khi gặp một lệnh nhánh, tầng decode chứa một chỉ thị nạp lệnh suy đoán có thể dẫn đến việc thực thi nhanh hơn. Bộ xử lý nạp lệnh dự định rẽ nhánh trong giai đoạn giải mã. Sau đó, trong giai đoạn thực thi, việc rẽ nhánh được giải quyết và bộ vi xử lý sẽ phân tích xem đâu là lệnh thực thi kế tiếp. Nếu việc rẽ nhánh không được chọn thì lệnh tiếp theo đã sẵn sàng. Còn nếu việc rẽ nhánh được chọn thì lệnh rẽ nhánh đó cũng đã sẵn sàng ngay lập tức, hạn chế thời gian rỗi chỉ còn một chu kỳ.  
Lõi Cortex-M3 chứa một bộ giải mã cho tập lệnh Thumb truyền thống và Thumb-2 mới, một ALU tiên tiến hỗ trợ nhân chia phần cứng, điều khiển logic, và các giao tiếp với các thành phần khác của bộ xử lý.  
Bộ vi xử lý Cortex-M3 là một bộ vi xử lý 32-bit, với độ rộng của đường dẫn dữ liệu 32 bit, các dải thanh ghi và giao tiếp bộ nhớ. Có 13 thanh ghi đa dụng, hai con trỏ ngăn xếp, một thanh ghi liên kết, một bộ đếm chương trình và một số thanh ghi đặc biệt trong đó có một thanh ghi trạng thái chương trình.  
Bộ vi xử lý Cortex-M3 hỗ trợ hai chế độ hoạt động (Thread và Handler) và hai mức truy cập tài nguyên của lõi xử lí (đặc quyền và không đặc quyền), tạo điều kiện cho việc cài đặt các hệ thống mở và phức tạp nhưng vẫn bảo mật. Những dòng mã không đặc quyền bị giới hạn hoặc không cho phép truy cập vào một số tài nguyên quan trọng (một số lệnh đặc biệt và các vùng nhớ nhất định). Chế độ Thread là chế độ hoạt động tiêu biểu hỗ trợ cả mã đặc quyền và không đặc quyền. Bộ vi xử lý sẽ vào chế độ Handler khi một ngoại lệ (exception) xảy ra và tất cả các mã là đặc quyền trong chế độ này. Ngoài ra, tất cả các hoạt động trong bộ vi xử lý đều thuộc một trong hai trạng thái hoạt động: Thumb cho chế độ thực thi bình thường và Debug cho việc gỡ lỗi.  
Bộ vi xử lý Cortex-M3 là một hệ thống ánh xạ bộ nhớ đơn giản, quản lí vùng nhớ cố định lên tới 4 gigabyte với các địa chỉ định nghĩa sẵn, dành riêng cho mã lệnh (vùng mã lệnh), SRAM (vùng nhớ), bộ nhớ/thiết bị bên ngoài, thiết bị ngoại vi bên trong và bên ngoài. Ngoài ra còn có một vùng nhớ đặc biệt dành riêng cho nhà cung cấp.  
****  
Hình 1.2. The Memory Map  
  
Bộ vi xử lý Cortex-M3 cho phép truy cập trực tiếp đến từng bit dữ liệu trong các hệ thống đơn giản bằng cách thực thi một kỹ thuật được gọi là bit-banding. Bộ nhớ bao gồm hai vùng bit-band (mỗi vùng 1MB) trong SRAM và vùng bí danh 32MB của vùng không gian ngoại vi (Mỗi byte trong vùng bí danh sẽ tương ứng với một bit trong vùng bit-band). Mỗi hoạt động nạp/lưu tại một địa chỉ trong khu vực bí danh (alias region) sẽ trực tiếp tương ứng với hoạt động trên bit được đại diện bởi bí danh đó. Cụ thể, khi ghi giá trị 0x01 vào một địa chỉ trên vùng bí danh thì có nghĩa là xác định bit tương ứng sẽ có giá trị là 1, tương tự giá trị 0x00 sẽ xác định bit tương ứng có giá trị 0. Còn đọc giá trị tại một địa chỉ vùng bí danh có nghĩa là đọc được giá trị của bit tương ứng. Một vấn đề cần chú ý nữa là hoạt động này mang tính nguyên tử (không chia nhỏ được nữa), không thể bị gián đoạn bởi các hoạt động khác trên bus.  
  
****  
Hình 1.3: Comparison of traditional bit manipulation with Cortex-M3- banding  
  
Các hệ thống cũ dựa trên ARM7 chỉ hỗ trợ truy xuất dữ liệu thẳng hàng, chỉ cho phép lưu trữ và truy xuất dữ liệu của một khối bộ nhớ mà mỗi phần tử có đơn vị là một word. Bộ vi xử lý Cortex-M3 hỗ trợ truy xuất dữ liệu không thẳng hàng, cho phép chuyển dữ liệu không thẳng hàng trong một truy xuất đơn. Thực tế, việc chuyển dữ liệu không thẳng hàng được biến thành việc chuyển nhiều lần dữ liệu thẳng hàng và có tính trong suốt đối với lập trình viên (nghĩa là lập trình viên hoàn toàn không cần quan tâm đến điều này). Ngoài ra bộ vi xử lý Cortex-M3 cũng hỗ trợ phép nhân 32-bit hoạt động trong một chu trình đơn và các phép chia có dấu, không dấu với các lệnh SDIV và UDIV, mất từ 2 đến 12 chu kỳ phụ thuộc vào kích thước của toán hạng. Phép chia được thực thi nhanh hơn nếu số chia và số bị chia có kích thước tương tự nhau. Những cải tiến trong khả năng toán học giúp Cortex-M3 trở thành bộ vi xử lý lý tưởng cho các ứng dụng thiên về tính toán như đọc cảm biến hoặc các hệ thống mô phỏng.

* 1. **Kiến trúc tập lệnh Thumb-2**

ARMv7-M là cấu hình vi điều khiển của kiến trúc ARMV7 và khác với các kiến trúc ARM trước đó ở chỗ nó chỉ hỗ trợ tập lệnh Thumb-2. Tập lệnh Thumb-2 là sự pha trộn giữa tập lệnh 16 và 32 bit, đạt được hiệu suất của các lệnh ARM 32 bit, đồng thời phù hợp với mật độ mã cũng như tương thích ngược với tập lệnh gốc Thumb 16 bit.  
  
****  
Hình 1.4: Relative Dhrystone performance and code size for ARM, Thumb, Thumb-2   
  
Trong một hệ thống dựa trên bộ vi xử lý ARM7, việc chuyển đổi nhân xử lý giữa chế độ Thumb (có lợi về mật độ mã) và ARM (có lợi về mặt hiệu suất) là cần thiết cho một số ứng dụng. Còn bộ vi xử lý Cortex-M3 có các lệnh 16 bit và 32 bit tồn tại trong cùng một chế độ, cho phép mật độ mã cũng như hiệu suất đều cao hơn mà không cần phải chuyển đổi phức tạp. Vì tập lệnh Thumb-2 là tập bao hàm của tập lệnh Thumb 16 bit nên bộ vi xử lý Cortex-M3 có thể thực thi các đoạn mã trước đây viết cho Thumb 16 bit. Do được cài đặt tập lệnh Thumb-2 nên bộ vi xử lý Cortex-M3 có khả năng tương thích với các thành viên khác của dòng ARM Cortex.  
Tập lệnh Thumb-2 có các lệnh đặc biệt giúp lập trình viên dễ dàng viết mã cho nhiều ứng dụng khác nhau. Các lệnh BFI và BFC là các lệnh thao tác trên bit, rất có ích trong các ứng dụng xử lý gói tin mạng. Các lệnh SBFX và UBFX giúp việc chèn vào hoặc trích xuất một số bit trong thanh ghi được nhanh chóng. Lệnh RBIT đảo bit trong một WORD, có ích trong các thuật toán DSP như DFT. Các lệnh bảng rẽ nhánh TBB và TBH tạo sự cân bằng giữa mật độ mã và hiệu suất. Tập lệnh Thumb-2 cũng giới thiệu cấu trúc If-Then  mới có thể xác định điều kiện thực hiện tối đa bốn lệnh tiếp theo.

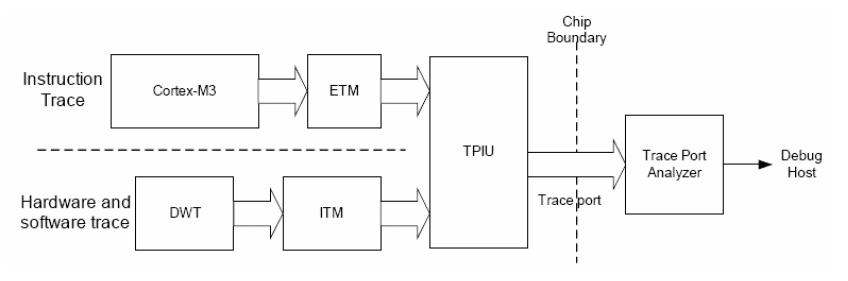
* 1. **Bộ điều khiển vector ngắt lồng nhau (****NVIC)**

NVIC (Nested Vectored Interrupt Controller) là thành phần tích hợp của bộ vi xử lý Cortex-M3 có khả năng xử lý ngắt rất linh hoạt và nhanh chóng. Trong cài đặt chuẩn, nó cung cấp một NMI (Non-Maskable Interrupt) và 32 ngắt vật lý đa dụng với 8 mức ưu tiên pre-emption. Nó có thể được cấu hình từ 1 đến 240 ngắt vật lý với tối đa 256 mức độ ưu tiên.  
Bộ vi xử lý Cortex-M3 sử dụng một bảng vector có thể tái định vị được, dùng để chứa địa chỉ của hàm xử lý ngắt. Khi nhận một ngắt, bộ xử lý sẽ lấy địa chỉ từ bảng vector thông qua bus chương trình. Bảng vector ngắt được đặt ở địa chỉ 0 khi reset, nhưng có thể được di chyển đến vị trí khác bằng cách lập trình một thanh ghi điều khiển.  
Để giảm bớt số cổng và tăng tính linh hoạt hệ thống, bộ vi xử lý Cortex-M3 đã chuyển từ mô hình ngoại lệ thanh ghi theo dõi của bộ vi xử lý ARM7 sang mô hình ngoại lệ dựa trên stack. Khi có một ngoại lệ xuất hiện thì bộ đếm chương trình (Program Counter), thanh ghi trạng thái chương trình (Program Status Register), thanh ghi liên kết (Link Register) và các thanh ghi đa dụng từ R0-R3, R12 bị đẩy vào ngăn xếp. Trong khi bus dữ liệu đẩy các thanh ghi lên vùng ngăn xếp thì bus chương trình xác định các vector ngoại lệ từ bảng vector và nạp lệnh đầu tiên của mã chương trình xử lí ngoại lệ. Sau khi hoàn tất việc lưu trữ dữ liệu trên ngăn xếp và nạp lệnh, chương trình phục vụ ngắt và xử lý lỗi được thực thi, tiếp theo đó các thanh ghi sẽ được phục hồi tự động để chương trình bị ngắt tiếp tục thực hiện bình thường. Vì thực hiện các hoạt động ngăn xếp bằng phần cứng nên ta không cần viết các đoạn hợp ngữ để thực hiện các thao tác trên ngăn xếp cho các hàm xử lý ngắt truyền thống dựa trên ngôn ngữ C, giúp việc phát triển ứng dụng dễ dàng hơn rất nhiều.  
NVIC hỗ trợ ngắt lồng nhau, cho phép một ngắt được xử lý trước một ngắt khác dựa trên mức độ ưu tiên. Nó cũng hỗ trợ cấu hình mức ưu tiên động cho các ngắt. Độ ưu tiên có thể được thay đổi bằng phần mềm trong thời gian chạy (run time). Các ngắt đang được xử lý đều bị khóa cho đến khi hàm xử lý ngắt hoàn thành, do đó, độ ưu tiên của ngắt có thể thay đổi mà không cần lo đến chuyện trùng lặp.  
Trong trường hợp các ngắt nối đuôi nhau, các hệ thống cũ sẽ lặp lại hai lần việc lưu trạng thái hoàn thành và khôi phục, dẫn đến độ trễ cao. Bộ vi xử lý Cortex-M3 đơn giản hóa việc chuyển đổi giữa các ngắt đang hoạt động và đang chờ bằng cách cài đặt công nghệ tail-chaining trong phần cứng NVIC. Tail-chaining đạt độ trễ thấp hơn nhiều bằng cách thay thế chuỗi các thao tác pop và push vốn mất hơn 30 chu kỳ xung nhịp bằng một thao tác nạp lệnh đơn giản chỉ mất 6 chu kỳ. Trạng thái bộ vi xử lý được tự động lưu khi ngắt bắt đầu được xử lý và phục hồi ngay khi kết thúc, ít chu kỳ hơn so với việc thực thi bằng phần mềm, nâng cao hiệu suất đáng kể ở hệ thống hoạt động dưới 100MHz.  
  
Hình 1.5. Tail chaining trong NVIC  
  
NVIC cũng cài đặt cách thức quản lý năng lượng của bộ vi xử lý Cortex-M3 tích hợp chế độ ngủ. Chế độ Sleep Now được gọi bằng một trong hai lệnh WFI (Wait For Interrupt) hoặc WFE (Wait For Event) sẽ ngay lập tức đặt nhân bộ vi xử lý vào trạng thái năng lượng thấp và chờ một ngoại lệ (exception). Chế độ Sleep On Exit đặt hệ thống vào chế độ năng lượng thấp ngay khi nó thoát khỏi hàm xử lý ngắt có độ ưu tiên thấp nhất. Nhân bộ vi xử lý vẫn ở trạng thái ngủ cho đến khi gặp một ngoại lệ. Vì chỉ có thể thoát khỏi chế độ này bằng ngắt nên trạng thái hệ thống không được phục hồi. Bit SLEEPDEEP của thanh ghi điều khiển hệ thống nếu được thiết lập có thể được sử dụng để khoá cổng (clock gate) lõi bộ vi xử lý và các thành phần hệ thống khác để tiết kiệm điện năng.  
NVIC cũng tích hợp một bộ đếm SysTick 24-bit đếm ngược (count-down timer) có thể được sử dụng để định thời tạo ra ngắt, cung cấp nhịp đập để một hệ điều hành thời gian thực hoạt động hoặc các tác vụ được lập lịch.

* 1. **Đơn vị bảo vệ bộ nhớ**

MPU là một thành phần tùy chọn của bộ vi xử lý Cortex-M3, có thể nâng cao độ tin cậy của hệ thống nhúng bằng cách bảo vệ các dữ liệu quan trọng được hệ điều hành sử dụng khỏi các ứng dụng khác, tách biệt độc lập các tác vụ đang thực thi bằng cách không cho phép truy cập vào dữ liệu của nhau, vô hiệu hoá quyền truy cập vào một số vùng nhớ, cho phép các vùng nhớ được định nghĩa là chỉ đọc (read only) và phát hiện các truy cập bộ nhớ có thể phá vỡ hệ thống.  
MPU cho phép một ứng dụng được chia nhỏ thành các tiến trình. Mỗi tiến trình sẽ có bộ nhớ (code, dữ liệu, ngăn xếp, heap) và thiết bị riêng, cũng như có quyền truy cập vào bộ nhớ và các thiết bị được chia sẻ. MPU cũng có các quy tắc (rule) truy cập của người dùng và đặc quyền bao gồm việc thực thi mã tại mức đặc quyền thích hợp cũng như quyền sở hữu bộ nhớ và các thiết bị của mã đặc quyền và mã người dùng.  
MPU chia bộ nhớ thành các vùng riêng biệt và thực hiện việc bảo vệ bằng cách ngăn các truy cập trái phép. MPU có thể chia bộ nhớ thành tối đa 8 vùng trong đó mỗi vùng có thể được chia thành 8 vùng con. Kích thước vùng có thể bắt đầu từ 32 byte và tăng gấp đôi dần cho đến  tối đa 4 gigabyte. Các vùng được đánh số thứ tự bắt đầu từ 0. Có thể xác định một bản đồ bộ nhớ (memory map) nền mặc định để truy cập đặc quyền. Việc truy cập đến các địa chỉ bộ nhớ không được xác định trong vùng MPU hoặc không được phép sẽ tạo ra ngoại lệ lỗi về quản lí bộ nhớ (Memory Management Fault Exception).  
Quy tắc bảo vệ vùng nhớ được dựa trên vào loại tác vụ (đọc, viết hoặc thực thi) và đặc quyền của mã thực hiện việc truy cập. Mỗi vùng bao gồm một bộ bit quy định loại truy cập được phép và hành động nào được phép trên bus. MPU cũng hỗ trợ các vùng chồng lên nhau (overlapping regions), tức là có sự giao nhau cùng một vùng địa chỉ. Vì kích thước mỗi vùng là bội số của 2 nên nếu 2 vùng chồng lên nhau thì sẽ có thể có một vùng nằm hoàn toàn trong vùng kia. Do đó, hoàn toàn có khả năng xảy ra trường hợp nhiều vùng nằm trọn trong một vùng hoặc trường hợp chồng lồng nhau. Trong trường hợp địa chỉ tra cứu nằm trong vùng chồng nhau thì kết quả trả về sẽ là vùng có số thứ tự cao nhất.

* 1. **Gỡ lỗi ( debug) và theo vết (trace)**

Việc gỡ lỗi hệ thống dựa trên bộ vi xử lý Cortex-M3 được thực hiện thông qua DAP (Debug Access Port), có thể là một cổng SWD (Serial Wire Debug) sử dụng 2 đường tín hiệu hoặc một cổng SWJ-D (Serial Wire JTAG Debug) sử dụng giao thức JTAG hoặc SW. Các SWJ-DP mặc định để chế độ JTAG khi reset và có thể chuyển giao thức với một chuỗi điều khiển cụ thể được cung cấp bởi phần cứng gỡ lỗi bên ngoài.  
Hành động debug có thể được kích hoạt bởi các sự kiện khác nhau như breakpoints, watchpoints, điều kiện lỗi hoặc yêu cầu debug từ bên ngoài. Khi một sự kiện debug xảy ra, bộ vi xử lý Cortex-M3 có thể vào chế độ tạm dừng (halt mode) hoặc chế độ theo dõi debug. Trong chế độ tạm dừng, bộ vi xử lí ngưng thực thi hoàn toàn các chương trình. Chế độ này hỗ trợ chạy từng bước. Lúc này, một ngắt phát sinh có thể bị trì hoãn đáp ứng, có thể được thực thi từng bước, hoặc bị che (masked) nên ngắt bên ngoài có thể bị bỏ qua trong quá trình debug. Trong chế độ theo dõi debug, một hàm xử lý ngoại lệ được thực thi để thực hiện việc gỡ lỗi trong khi vẫn cho phép các exception có độ ưu tiên cao hơn diễn ra. Chế độ này cũng hỗ trợ chạy từng bước.  
Bộ FPB (Patch Flash and Breakpoint ) có 6 breakpoint trong chương trình và 2 breakpoint nạp dữ liệu, hoặc chuyển lệnh/dữ liệu từ bộ nhớ mã đến bộ nhớ hệ thống. Bộ FPB này có sáu comparator để so sánh các lệnh được lấy từ bộ nhớ mã. Mỗi comparator có thể được kích hoạt để định vị lại mã chương trình đến một vùng trong bộ nhớ hệ thống, hoặc thực hiện một breakpoint phần cứng bằng cách trả về một lệnh breakpoint cho bộ vi xử lý. Nó cũng có hai comparator với nhiệm vụ tương tự cho dữ liệu.  
Bộ DWT (Data Watchpoint and Trace) có bốn comparator có thể được cấu hình thành watchpoint phần cứng. Khi được sử dụng trong cấu hình này, comparator có thể được lập trình để so sánh địa chỉ truy cập dữ liệu hoặc bộ đếm chương trình. Các DWT comparator cũng có thể được cấu hình để kích hoạt các sự kiện lấy mẫu PC, sự kiện lấy mẫu địa chỉ dữ liệu và làm cho ETM (Embedded Trace Macrocell) phát ra các gói kích hoạt trong dòng lệnh đang được truy vết.  
ETM là một thành phần tùy chọn để hỗ trợ việc theo vết lệnh để đảm bảo rằng có thể tái cấu trúc lại việc thực hiện chương trình mà chỉ ảnh hưởng một cách tối thiểu đến bộ nhớ. ETM cho phép truy vết theo thời gian thực về việc thực thi lệnh và truyền dữ liệu bằng cách nén thông tin truy vết từ nhân bộ xử lý để giảm thiểu yêu cầu băng thông.  
  
Hình 1.6 Hệ thống theo vết Cortex-M3  
  
Bộ vi xử lý Cortex-M3 thực hiện việc theo vết dữ liệu bằng DWT và ITM (Trace Instrumentation Macrocell). DWT cung cấp số liệu thống kê về việc thực hiện lệnh và có thể tạo ra sự kiện watchpoint để gọi debug hoặc ETM trên các sự kiện hệ thống. ITM là công cụ truy vết ứng dụng hỗ trợ cách gỡ lỗi kiểu "printf" cho hệ điều hành và theo vết các sự kiện ứng dụng. Nó chấp nhận các gói truy vết phần cứng từ DWT và phần mềm theo dõi sự kích thích từ lõi bộ vi xử lý và phát ra thông tin chẩn đoán hệ thống định kỳ. Bộ TPIU (Trace Port Interface Unit) chấp nhận các thông tin truy vết từ ETM và ITM, sau đó hòa trộn chúng, định dạng lại và phát ra thông qua SWV (Serial Wire Viewer) đến các bộ phân tích truy vết bên ngoài. SWV cho phép tạo ra profile cho các sự kiện hệ thống một cách đơn giản và hiệu quả bằng cách xuất dòng dữ liệu thông qua một pin duy nhất. Mã hóa Manchester và UART là các định dạng được hỗ trợ cho SWV.

* 1. **Ma trận bus và các giao diện liên kết**

Ma trận bus của bộ vi xử lý Cortex-M3 kết nối bộ xử lý và giao diện debug đến các bus bên ngoài, ICode, DCode và giao diện hệ thống dựa trên AMBA AHB-Lite 32 bit, và bus cho các ngoại vi (Private Peripheral Bus) dựa trên AMBA APB 32 bit. Ma trận bus cũng đảm nhiệm việc truy cập dữ liệu không thẳng hàng và các vùng bit-banding.  
Giao diện ICode 32 bit lấy các lệnh từ vùng nhớ chương trình và chỉ có thể truy cập bởi CM3Core. Tất cả các lần nạp lệnh đều có độ rộng là một từ (WORD), với số lượng lệnh được lấy trên mỗi từ tùy thuộc vào loại mã thực hiện và vị trí của nó trong bộ nhớ. Giao diện DCode 32 bit truy cập dữ liệu từ vùng nhớ mã chương trình và có thể được truy cập bởi CM3Core và DAP. Giao diện hệ thống 32 bit lấy các lệnh và truy cập dữ liệu trong vùng bộ nhớ hệ thống và giống như bus DCode, có thể được truy cập bởi CM3Core và DAP. PPB cho phép truy cập vào các thành phần bên ngoài của hệ thống Cortex-M3.

# Chương 2: NEC Infrared Transmission Protocol

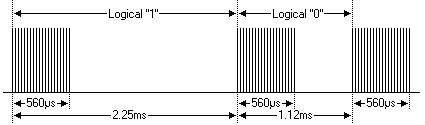
1. **NEC protocol**

Tất cả những khái niệm, kiến thức căn bản về giao thức dưới đây được phát triển bởi NEC nay gọi là Renesas, giao thức này còn được gọi là Japanese Format.  
Cho đến nay, vẫn chưa có một thông tin cụ thể, chính xác về người đã phát triển chuẩn giao thức này, nó được sử dụng trong VCR sản xuất bởi công ty Sanyo và được thương mại hóa ra thị trường với cái tên Fisher, và nhà máy của NEC đã chế tạo ra IC điều khiển từ xa đầu tiên.  
Những kiến thức, mô tả căn bản về NEC được lấy trong tài liệu RM của VCR và tài liệu này ngày càng được bổ sung, cập nhật những thông tin , mô tả hữu ích.

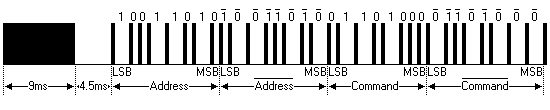
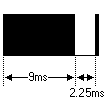
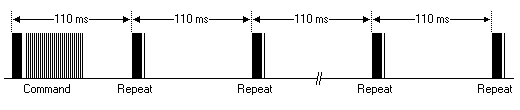
1. **Đặc điểm**

- Gồm 8 bit địa chỉ và 8 bit mã lệnh  
**-** Chế độ mở rộng sẵn có, gấp đôi số bit địa chỉ.  
- Địa chỉ và mã lệnh được truyền đi 2 lần để tăng độ tin cậy.  
- Điều chế độ rộng xung   
- Tần số sóng mang là 38 kHz  
- Bit time: 1,125ms hoặc 2.25ms

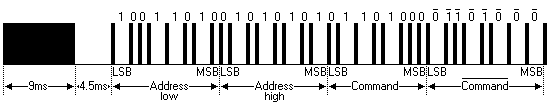
1. **Điều chế**

****Hình 2.1 Chuẩn giao thức NEC sử dụng mã hóa độ rộng xung của các bit.   
Mỗi xung khoảng 560 micro giây với sóng mang 38kHz ( khoảng 21 chu kỳ máy).  
Mức logic 1 mất 2,25 ms cho qua trình truyền   
Mức logic 0 mất 1,125 ms cho quá trình truyền   
Theo đó, mức logic 0 chỉ mất 1 nửa thời gian truyền của mức logic 1.   
Mức khuyến nghị ở chu kỳ sóng mang ¼ hoặc 1/3.

1. **Giao thức**

****Hình 2.2 Dãy xung của giao thức NECHình ảnh trên cho thấy kiểu dãy xung của giao thức NEC.  
Với giao thức này thì LSB sẽ được truyền đi đầu tiên. Trong trường hợp này địa chỉ và mã lệnh sẽ được truyền đi. Bản tin sẽ được bắt đầu bằng 9ms AGC. Sau AGC là 4,5 ms trống.  
Theo sau 4,5 ms trống sẽ là dữ liệu chính của package dữ liệu gồm địa chỉ và mã lệnh thực hiện   
Như đã nói, địa chỉ, mã lệnh của bản tin được truyền đi 2 lần để đảm bảo tính chính xác, độ ổn định, độ tin cậy.   
Trong lần truyền thứ 2, các bít sẽ được đảo lại để kiểm tra, kiểm định tính chính xác của bản tin đã nhận được ở lần 1.  
Tổng thời gian truyền tin luôn là hằng số vì các bít được truyền lại lần 2 với dạng nghịch đảo.  
Trong những trường hợp cần tới tốc độ cao mà bỏ qua tính chính xác hay độ tin cậy của mạch thì có thể bỏ đi lần truyền bit thứ 2 này. Nghĩa là các bit nghịch đảo trong lần 2 không được truyền đi, quá trình kiểm định tính chính xác dựa vào bit nghịch đảo sẽ không còn. Đảm bảo tốc độ truyền nhanh nhưng tăng khả năng lỗi bit, gây sai lệch quá trình truyền nhận điều khiển.  
Một cách khác có thể tăng tốc độ truyền là mở rộng địa chỉ và mã lệnh lên 16 bit. Điều này tương tự như trên, cũng sẽ làm tốc độ truyền tăng lên, tỷ lệ bit lỗi cao hơn  
Kết hợp lại, khi tăng địa chỉ, mã lệnh lên 16 bit, đồng thời bỏ quá trình truyền bit đảo để kiểm tra lỗi sẽ đưa ra tốc độ cao nhất nhưng tỷ lệ lỗi cũng lớn nhất.   
Việc cân nhắc lựa chọn thuật toán và xử lý bit là rất quan trọng.  
  
****  
Hình 2.3 LBS  
  
Mã lệnh chỉ được truyền đi 1 lần duy nhất cho dù là phím trên remote vẫn đang được nhấn.  
Trong trương hợp nhấn giữ, cứ 110ms, code lại được truyền đi chừng nào phím vẫn được giữ xuống. Repeat code chỉ đơn giản là 9 ms xung AGC được theo sau bởi 2.25ms trống và 560 micro giây ngắt gián đoạn.  
  
****  
  
Hình 2.4 Repeat code

1. **Giao thức mở rộng NEC**

Chuẩn giao thức NEC được sử dụng rất rộng rãi. Bằng việc bỏ đi các địa chỉ dư thừa, địa chỉ có thể được mở rộng từ giá trị dương 256 tới xấp xỉ 65000. Đây chính là cách địa chỉ có thể được mở rộng từ 8 bit lên 16 bit mà không cần phải thay đổi bất kỳ đặc tính kỹ thuật nào của giao thức.  
Việc mở rộng độ dài của địa chỉ giúp tổng thời gian truyền tin không hề lâu hơn hằng số.   
Theo như mô tả, thời gian truyền đi mức logic 0 và mức logic 1 là khác nhau nên, tốc độ truyền tin thực chất còn quyết định bởi nội dung của bản tin. Rõ ràng, với một bản tin nhiều bit 0 hơn sẽ có thời gian truyền tin ngắn hơn bản tin nhiều bit 1 hơn, qua đó thời gian, tốc độ truyền tin cũng thay đổi theo.  
Nếu muốn giữ thời gian truyền tin là cố định, phải đảm bảo rằng số bit 1 phải bằng số bit 0, cũng có nghĩ là số bit 1 trong địa chỉ phải là 8.   
Điều đó sẽ làm giản số lượng địa chỉ tạo ra xuống còn 13000.  
Những mã lệnh dư thừa vẫn được giữa nguyên. Chính vì vậy, mỗi địa chỉ phải điều khiển 256 mã lệnh khác nhau.  
  
****  
  
Hình 2.5 Address-16bit  
  
Thêm vào đó, phải luôn ghi nhớ rằng 256 giá trị địa chỉ có thể của giao thức mở rộng luôn phải hợp lệ vì đó thực sự là những địa chỉ của giao thức NEC. Cứ khi nào những byte thấp lại là nghịch đảo của những byte cao thì đó không phải là địa chỉ hợp lệ

# Chương 3: Hồng ngoại và ứng dụng

Tia hồng ngoại là bức xạ điện từ có bước sóng dài hơn ánh sáng nhìn thấy nhưng ngắn hơn tia bức xạ vi ba. Tên “hồng ngoại” có nghĩa là “ngoài mức đỏ”, màu đỏ là màu có bước sóng dài nhất trong ánh sáng thường.

|  |  |  |  |
| --- | --- | --- | --- |
| Tên | Bước sóng | Tần số | Năng lượng |
| Tia gamma | <0,01 nm | >= 30 EHz | 124 keV – 300 GeV |
| Tia X | 0,01 – 10 nm | 30 EHz – 30 PHz | 124 eV – 124 keV |
| Tia tử ngoại | 10 – 380 nm | 30 PHz – 790 THz | 3.3 eV -124 eV |
| Ánh sáng nhìn thấy | 380 – 700 nm | 790 THz – 430 THz | 1.7 eV – 3.3 eV |
| Tia hồng ngoại | 700 – 1mm | 430 THz – 300 GHz | 1.24 meV – 1.7 eV |
| Vi ba | 1mm – 1 m | 300 GHz – 300 MHz | 1.7 eV – 1.24 meV |
| Radio | 1mm – 100000 km | 300 GHz – 3 Hz | 12.4 feV – 1.24 meV |

Bảng 3.1 Bảng phân chia các bức xạ sóng điện từ/ ánh sáng

Vùng ánh sáng mà mắt người thông thường nhìn thấy, được áp đặt gọi là ánh sáng nhìn thấy, có bước sóng 0,38 – 0,7 um hay tần số trong dải 430-790 THz. Bức xạ hồng ngoại được định nghĩa có bước song từ 0,7 um – 1 mm. Một số sinh vật có thể nhìn thấy tia hồng ngoại ở vùng gần kề với ánh sáng thường, cũng như trong một số thí nghiệm thì có người nhìn thấy đến vùng hồng ngoại 1,05 um

1. **Phân loại**

Tia hồng ngoại được phân chia theo bước song thành ba vùng chính

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên | Viết tắt | Bước sóng | Tần số | Năng lượng photon | Đặc trưng |
| Hồng ngoại gần | NIR, IR-ADIN | 0,75-1,4 um | 214-400 THz | 886-1653 meV | Được xác định bởi sự hấp thụ của nước, và thường được sử dụng trong vieenc thông sợi quang vì tỏn thất do suy giảm trong thủy tinh SiO2 là ở mức trung bình. Các máy khuếch đại hình ảnh rất nhạy cảm với vùng quang phổ này, như trong các thiết bị nhìn đêm |
| Hồng ngoại sóng ngắn | SWIR, IR-BDIN | 1,4-3 um | 100-214 THz | 413-886 meV | Hấp thụ trong nước tăng đáng kể tại 1,45 um. Dải 1,53-1,56 um là vùng phổ hiện dùng nhiều trong viễn thông đường dài |
| Hồng ngoại sóng trung | MWIR, IR-CDIN;MidIR. | 3-8 um | 37-11 THz | 155-413 meV | Trong công nghệ dẫn đường tên lửa thì vùng 3-5 µm là cửa sổ khí quyển, trong đó "đầu dò tầm nhiệt" IR thụ động của tên lửa được bố trí để làm việc, dẫn đường vào chỉ dấu hồng ngoại của máy bay mục tiêu, thường là chùm ống xả của động cơ phản lực. Dải này còn được gọi là hồng ngoại nhiệt, nhưng nó chỉ phát hiện được nhiệt độ hơi cao hơn nhiệt độ cơ thể. |
| Hồng ngoại sóng dài | LWIR, IR-C*DIN* | 8-15 um | 20-37 THz | 83-155 meV | Vùng của các ảnh nhiệt trong đó các cảm biến có thể hoàn toàn thụ động thu được hình ảnh các đối tượng có nhiệt độ chỉ hơi cao hơn nhiệt độ phòng, ví dụ cơ thể con người, mà không cần ánh sáng chiếu vào từ mặt trời, mặt trăng, hoặc đèn chiếu hồng ngoại. Vùng này còn được gọi là "hồng ngoại nhiệt". |
| Hồng ngoại xa | FIR | 15-1000 um | 0.3-20 THz | 1.2-83 meV | Xem hồng ngoại xa và laser hồng ngoại xa |

Bảng 3.2 Bảng phân loại tia hồng ngoại thông dụng theo phân loại Mỹ

1. **Nguồn phái tia hồng ngoại**

Mọi vật có nhiệt độ lớn hơn 00 K đều phát ra tia hồng ngoại

1. **Tính chất**

Tia hồng ngoại có các tính chất cơ bản sau:  
- tác dụng nhiệt  
- có thể gây ra hiện tượng quang điện trong ở chất bán dẫn  
- có thể tác dụng lên một số kính ảnh đặc biệt  
- có thể biến điệu như song điện từ cao tần

1. **Ứng dụng**
   1. **Đo nhiệt độ**

Việc thu nhận và đo đạc tia hồng ngoại có thể giúp xác định nhiệt độ của vật từ xa, nếu chúng là nguồn phát ra các tia thu được. Hình chụp trong phổ hồng ngoại được gọi là hình ảnh nhiệt, hay trong trường hợp vật rất nóng trong NIR hay có thể thấu được gọi là phép đo nhiệt.  
Kỹ thuật đo nhiệt độ bằng hồng ngoiaj được dùng chủ yếu trong quân sự, và ứng dụng công nghiệp. Kỹ thuật này hiện cũng đang được ứng dụng và dần quen thuộc với thị trường dân sự như: máy ảnh trên xe hơi, tùy thuộc vào giá thành của các sản phẩm có được giảm giá mạnh hay không.

* 1. **Phát nhiệt**

Tia hồng ngoại được dùng trong đèn hồng ngoại sưởi trực tiếp lên cơ thể, và bố trí ở một số phòng tắm hơi. Tuy nhiên cần lưu ý không nhìn vào các đèn này vì mắt không điều tiết được độ mở sáng theo tia hồng ngoại, chúng có thể gây mù mắt. Tia hồng ngoại có thể được dùng làm tan tuyết trên cánh máy bay.  
Một lượng lớn năng lượng mặt trời là nằm trong vùng hồng ngoại. Các vật nóng cỡ vài trăm độ C như lò sưởi, bếp cũng phát ra bức xa vật đen có cực đại ở vùng hồng ngoại. Do vậy tia hồng ngoại còn được gọi là tia nhiệt.

* 1. **Kỹ thuật hồng ngoại trong quân sự**

Kỹ thuật hồng ngoại rất quan trong với ngành quốc phòng. Những tên lửa không đối không cự ly gần mà máy bay chiến đấu sử dụng đều có dùng tia hồng ngoại dẫn đường, thường được gọi là tên lửa tầm nhiệt hay tên lửa dẫn hướng hồng ngoại. tên lửa tự động bám sát luồng hợi nóng từ động cơ máy bay để tìm đến địch.  
Để chống lại tên lửa tầm nhiệt thì máy bay bố trí các quả pháo nòng sáng, tung ra khi phát hiện có tên lửa. Nó dẫn đến cuộc đua, một mặt là tăng khả năng nhận dạng bằng ảnh hồng ngoại cho tên lửa, mặt khác là sử dụng cùng với các dạng dẫn hướng khác.

* 1. **Truyền thông**

Tia hồng ngoại gần và trung được dùng trong viễn thông cáp quang do có tổn hao nhỏ, cũng như do công nghệ chế tạo linh kiện phát và thu tín hiệu quy định  
.

|  |  |  |
| --- | --- | --- |
| Băng | Mô tả | Dải sóng |
| O band | Ỏriginal | 1260 – 1360 |
| E band | Extended | 1360 – 1460 |
| S band | Short wavelength | 1460 – 1530 |
| C band | Conventional | 1530 – 1565 |
| L band | Long wavelength | 1565 – 1625 |
| U band | Ultralong wavelength | 1625 – 1675 |

Bảng 3.3 Bảng băng tần viễn thông IR

* 1. **Thiết bị nhìn đêm**

Thiết bị nhìn đêm là thiết bị quang học- điện tử thực hiện quan sát được môi trường quan tâm trong điều kiện đêm tối hay những khi vực có ánh sáng yếu. Thiết bị thu nhận tia hồng ngoại bằng các ống kính quang học và hiện hình ảnh trên màn hình điện tử. Chúng được chia thành 3 loại chính:   
- Thiết bị hồng ngoại gần bị động. Thu nhận ánh sáng ở vùng nhìn thấy và hồng ngoại gần rồi khuếch đại, cho ra ảnh đơn sắc. Chúng có tầm quan sát xa, và được chế thành kính ngắm bắn tỉa, kính nhìn đêm trong quân sự cho phi công, xe tăng biệt kích, điều tra viên, …  
- Thiết bị hồng ngoại nhiệt bị động: là camera quan sát thu nhận ánh sáng ở vùng hồng ngoại nhiệt hay hồng ngoại xa, hiện trên màn hình theo thang nhiệt độ thiên nhiên. Chúng được dùng trong trinh sát, bảo vệ, … hoặc trong nghiên cứu hoạt động ban đêm của động vật, như trong ảnh minh họa chú chó.  
- Thiết bị hồng ngoại gần chủ động, là camera quan sát hồng ngoại có gắn kèm LED phát bức xạ hồng ngoại, tức là chủ động về nguồn chiếu sáng nhưng không để cho mắt thường nhìn thấy. Chúng có tầm quan sát gần và được sử dụng phổ biến trong hoạt động dân sự, như trong thị trường Việt Nam hiện nay.  
Trong hoạt động thực tế có thể dùng đến đèn pha hồng ngoại chiếu sáng vùng quan sát để thu nhận hình ảnh rõ hơn, như trong hoạt động bảo vệ biên giới ở Mỹ chống lại người di cư lậu.

* 1. **Nghiên cứu thiên văn**

Trong thiên văn học quan sát hồng ngoại đặc biệt có ý nghĩa trong phát hiện và nghiên cứu các đối tượng “lạnh” có nhiệt độ dưới 1000 độ K, và khó có thể nhìn thấy trong vùng quang phổ khác, hoặc các đối tượng ở trong hoặc phía sau một đám mây liên sao.  
Ngoài ra, quan sát phổ hồng ngoại được dùng trong phân tích đặc điểm của các đối tượng bất kỳ. Một số vật chất ở các sao được phát hiện nhờ vào quang phổ hồng ngoại, ví dụ, phát hiện khí metan trên hành tinh của hệ ngôi sao cố định HD 189733.

* 1. **Bảo mật tiền và tài liệu quý**

Bảo mật tài liệu dùng tia hồng ngoại thực hiện cho tài liệu quan trong như hộ chiếu, tiền hay chứng chỉ ngân hang, … Tùy theo mắc bảo mật mà khi chế tạo giấy được trộn chất có phản ứng xác định với dải tia hồng ngoại nhất định. Ở mức phức tạp thì xếp đặt chất đó theo ký hiệu xác định. Các máy kiểm tra dùng đèn hồng ngoại có khoảng phổ đã thiết kế chiếu lên giấy sẽ làm rõ những yếu tố bảo mật có hay không.  
Bảo mật này đang được áp dụng cho đồng EUR. Hộ chiếu Anh thì dùng huỳnh quang hồng ngoại của methylene xanh. Tuy nhiên việc dùng tia hồng ngoại dễ gặp lỗi quan sát hơn so với việc dùng tia tử ngoại.

* 1. **Điện tử điều khiển**
     1. **Tự động bật tắt thiết bị**

Tại các nơi công cộng như sân bay, nhà ga, cửa hang, bệnh viện, nhà riêng, … thì việc tự động đóng mở cửa, bật đèn, vòi nước, được thực hiện bằng cảm biến hồng ngoại ( mắt thần ) nhận biết người hoặc vật chuyển động thông qua nhiệt độ cao hơn xung quanh. Tuy nhiên nếu chỉ dùng cảm biến hồng ngoại thì hoạt động cảm biến dễ lỗi khi nhiệt độ môi trường cao hơn 35 độ.  
Các mắt thần này dùng diode quang loại tiếp nhận hồng ngoại để cảm biến. Mắt dùng 1 diode thì cảm nhận gần và hẹp, mắt dùng nhiều diode thì mỗi diode giám sát một góc đặc nhất định, và nâng khoảng cách cảm nhận đến 3-5m.

* + 1. **Phụ kiện thiết bị vi tính**

Một số chuột quang cũng dùng tia hồng ngoại, tuy nhiên chuột này cần có thêm LED báo có cấp nguồn.  
Tia hồng ngoại cũng được dùng để truyền tải thông tin trong mạng nhỏ, ví dụ như từ máy tính sang máy tính, máy tính sang điện thoại, điện thoai với điện thoại… hoặc các thiết bi gia dụng khác. Tuy nhiên khoảng cách truyền ngắn và dễ nhiễu.

* + 1. **Các điều khiên xa (remote control)**

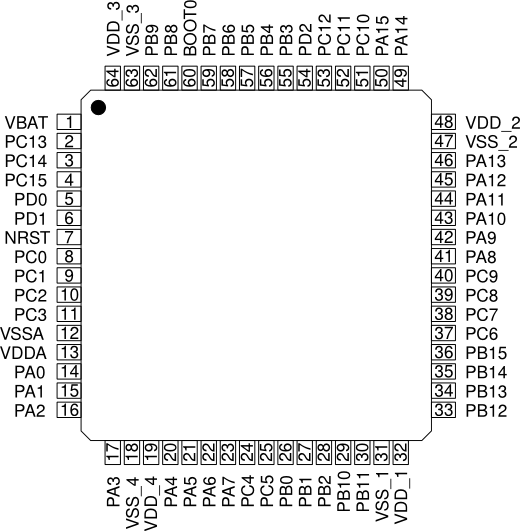
Các điều khiển xa, thường gọi là remote control, phần lớn dùng tia hồng ngoại để điền khiển TV, dàn âm thanh, hình ảnh, quạt …   
Một biến thể dân dụng ăn thiều là đèn chiếu sáng LED điều khiển từ xa hồng ngoại, bật tắt bằng bất kỳ remote control nào trong các loại trên. Sự khám phá ra tia hồng ngoại thường được cho là công lao của William Herschel, nhà thiên văn học đầu thế kỷ 19. Herschel dùng lăng kính để tán xạ ánh sáng từ Mặt trời và khám phá ra tia hồng ngoại, nằm ngoài vùng ánh sáng nhìn thấy gần phần ánh sáng đỏ, thông qua sự ghi chép trên một nhiệt kế

# Chương 4: Thiết kế mạch điều khiển từ xa bằng hồng ngoại

1. **Giới thiệu**
   1. **Vi điều khiển STM32F103**

Hình 4.1 Vi điều khiển STM32F103Vi điều khiển STM32F103XX được sản xuất bởi ST company và ứng dụng trong đồ án này. Công ty Baltic Engineering quyết định sử dụng vi điều khiển STM32F103 trong phần lớn các dự án vì vi điều khiển này có rất nhiều tính năng cao cấp so với dòng vi điều khiển khác. Đây là một trong số những vi điều khiển tốt nhất trong dòng MCU 32-bit, hiệu suất cao nhất để điều khiển và kế nối trong các dự án điện tử, nó có thể dùng trong các giải pháp xử lý tín hiệu số ( nhờ vào xung nhịp cao ), ứng dụng năng lượng thấp để tiết kiệm năng lượng cho hệ thống, tốc độ của các ngoại vi tăng lên khi hiệu suất cao hơn.

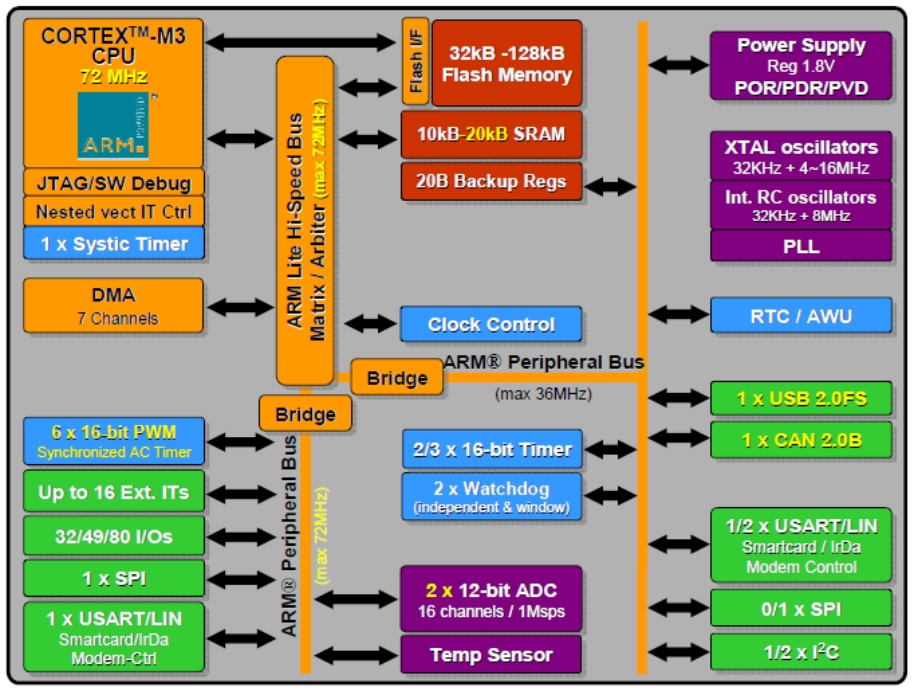
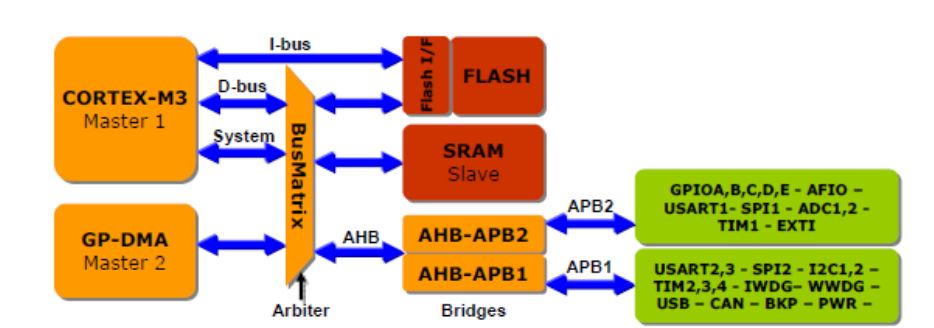
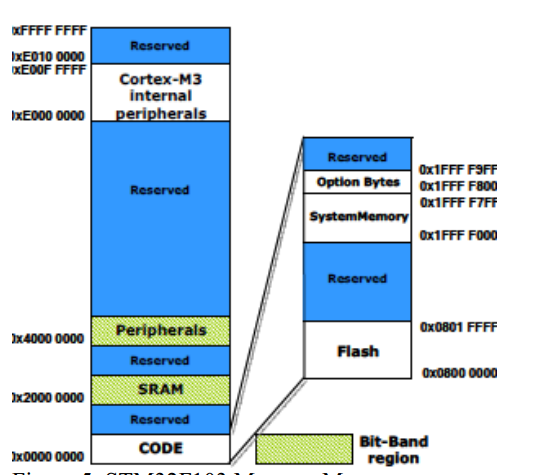
* + 1. **Tổng quan**

Họ vi điều khiển của STM32F103xx bao gồm lõi ARM Cortex-M3 32-bit, bộ nhớ nhúng tốc độ cao ( Bộ nhớ Flash lên tới 128 Kbyte và SRAM lên tới 20Kb), I/Os và các ngoại vi được hoạt động đông thời nhờ vào kết nối đến 2 buses APB. Vi điều khiển STM32F103xx gồm rất nhiều ngoại vi như 2 ADC 12-bit, 1 bộ điểu khiển Timer cao cấp, 3 GP Timer 16 bit và có Timer PWM. Nó cũng mang đến 2 I2C và SPI, 3 USART, 1 USB và 1 CAN là giao thức truyền thông.   
Sơ đồ chân của STM32F103xx được sử dụng trong đề tài được mô tả chi tiết trong hình 4.2.  
   
Hình 4.2 STM32F103xx pinout  
  
Họ vi điều khiển này chứa 3 port là PA,PB,PC là các port của MCU, mỗi port có 16 chân I/O. Vss, Vss và Vbat được dùng để tính sai số của vi điều khiển bằng cách dùng điện áp đầu vào tham chiếu.

* + 1. **Họ vi điều khiển của STM32F103xx**

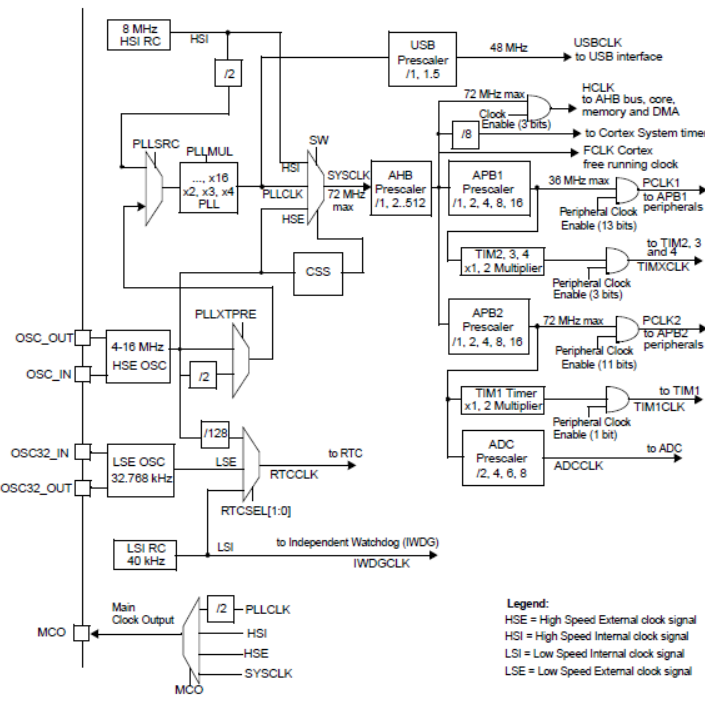
Họ vi điều khiển STM32F103xx được chia thành 3 nhóm:  
Low-density: STM32F103x4 và STM32F103x6 là các thiết bị low-density.  
Medium-density: STM32F103x8 và STM32F103xB là các thiết bị medium-density.  
High-density: STM32F103xC, STM32F103xD và STM32F103xE là các thiết bị high-density.

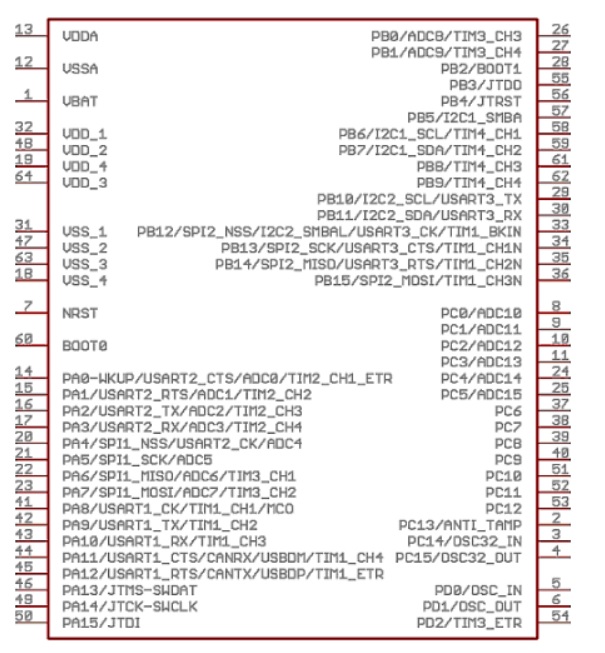
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Pinout | Low-denstisy | | Medium-density | | High-density | | |
| 16 KB Flash | 32 KB Flash | 64 KB Flash | 128 KB Flash | 256 KB Flash | 384 KB Flash | 512 KB Flash |
| 6 KB RAM | 10 KB RAM | 20 KB RAM | 20 KB RAM | 48 KB RAM | 64 KB RAM | 64 KB RAM |
| 144 |  |  |  |  | 5 x USART  4 x 16-bit timers, 2 x basic timers  3 x SPI, 2 x I2S, 2 x I2C USB, CAN, 2 x PWM timer 3 x ADC, 1 x DAC, 1 x SDIO  FSMC ( 100 và 144 chân ) | | |
| 100 |  |  | 3 x USART  3 x 16-bit timer  2 x SPI, 2 x I2C, USB, CAN, 1 x PWM timer, 2 x ADC | |
| 64 | 2 x USART  2 x 16-bit timer  1 x SPI, 1 x I2C, USB, CAN, 1 x PWM timer  2 x ADC | |
| 48 |  |  |  |
| 36 |  |  |  |  |  |

Bảng 3.4 Phân loại các nhóm theo chức năng STM32F103xx  
Ba nhóm trên được chia theo chức năng của họ vi điều khiển STM32F103xx  
Vi điều khiển low-dentisy có bộ nhớ Flash thấp hơn và RAM, ít timer và ngoại vi hơn 2 nhóm còn lại. Vì vậy, Medium-density và High-dentisy bao gồm bộ nhớ Flash cao hơn, RAM tốt hơn và có nhiều ngoại vi mở rộng hơn.  
Dòng Low-density bao gồm 16 KB đến 32 KB bộ nhớ Flash và 6 KB đến 10 KB RAM. Nó chứa 1 CAN, 1 USB, 1 PWM timer, 1 I2C, 1 SPI và 2 ADC, 2 USART , 2 timer 16 bit. Sự khác nhau giữa các dòng Low-density là số lượng gói chân ra của nó. Có 3 loại gói chân ra là 36,48,64 chân. Bộ nhớ flash cũng tăng từ 64 KB đến 128 KB và RAM 20 KB.  
Dòng Medium-density có nhiều đặc tính hơn so với dòng low-density. Số lượng ngoại vi và chân ra của chúng được tăng lên. Có tất cả 1 CAN, 1 USB, 1 PWM timer, 2 I2C, 2 SPI, 2 ADC, 3 USART, 3 timer 16 bit. Nó có tổng cộng 3 loại gói chân gồm 48,64,100 chân.   
Dòng High-density thì hơn hoàn toàn các dùng khác và có nhiều ngoại vi hơn. Vi dụ như: 1 × CAN, 1 × USB, 1 × PWM timer, 2 × I²Ss (SPI), 2 × I²C, 10 3 × SPI, 2 × ADCs, 1 × DAC, 5 × USARTs, 2 × basic timers, 4 × 16-bits timers, 1 × SDIO (Secure Digital Input Output), and 1 ×FSMC (Flexible Static Memory Controller).   
Vi điều khiển STM32F103xx thực sự thuận tiện để sử dụng cho rất nhiều các ứng dụng khác nhau như trong công nghiệp, điều khiển, thiết bị y tế, Thiết bị chơi game trên PC, GPS, Liên lạc qua video, hệ thống cảnh báo …  
Cấu trúc của MCU STM32F103  
Core của vi điều khiển, bộ nhớ, chân IO và các ngoại vi được mô tả trong sơ đồ hình 4.3:  
  
  
Hình 4.3 STM32F103 Performance line Block Diagram  
  
Cấu trúc hệ thống bao gồm Buses, DMA, SRAM nội, Bộ nhớ Flash nội, một vài trong số đó được cân nhắc làm master và còn lại là slave.   
Hình 4.4 Mô tả cấu trúc bus cho họ vi điều khiển STM32. Việc truy cập giữ bus lõi hệ thống và bus DMA được điều khiển bởi BusMatrix.  
  
  
Hình 4.4 System Architecture   
  
Có 4 khối master và 3 khối slave trong cấu trúc được nói đến ở trên  
  
Master:  
I-bus : Kết nối lõi Cortex M3 tới cấu trúc bộ nhớ flash để thực thi.  
D-bus: Kết nối lõi M3 đến bộ nhớ flash về dữ liệu giao diện  
S-bus: kết nối bus ngoại vi của M3 đến BoxMatrix để điều khiển thành phần ản giữa DMA và Lõi.  
GP-DMA: Kết nối CPU và DMA đến bộ nhớ flash, SRAM và ngoại vi thông qua BoxMatrix để tạo ra kết nối giữa chúng.  
  
Slave:  
SRAM nội  
Bộ nhớ Flash nội  
AHB đến APB: cầu chia bus AHP vào 2 bus, APB1,APB2. APB1 để dùng cho ngoại vi mà có tần số 36 MHz và APB2 dùng cho ngoại vi chạy trên 72 MHz.  
  
**Lõi ARM M3** là CPU của vi điều khiển và là 1 trong những phần quan trọng nhất của của vi điều khiẻn. Lõi này là bản mới nhất của vi xử lý ARM được ứng dụng cho hệ thống nhúng. Nó hoạt động nhanh nhất trên xung nhịp 72 MHz, 90DIMPS với 1.25 DIMPS/MHz, hiệu suất tại trạng thái truy cập bộ nhớ 0.   
Hệ thống nhớ của vi điều khiển bao gồm 2 phần là nhớ flash và nhớ SRAM. Bộ nhớ flash để lưu trữ dữ liệu và chương trình và có thể nâng lên thành 128 Kb. Bộ nhớ SRAM là bộ nhớ đọc nghi của CPU với trạng thái đợi 0 để lưu trữ dữ liệu cho việc xử lý bởi USB và có thể lên tới 20 Kb.  
*NVIC*: Nó có thể được dùng để điều khiển lên tới 43 kênh ngắt ( là những ngắt đặc biệt có thể enable/disable hay điều khiển bởi chương trình) và, nó có 16 mức ưu tiên lập trình. Nó được dùng để đặt ưu tiên kênh IRQ.  
*EXTI* : nó có 19 lines bắt sườn để tạo ngắt/ sự kiện yêu cầu. Để bắt được trigger ngoại có thể sử sụng, có thể để bắt thể bắt theo sườn lên hoặc sườn xuống hoặc cả lên và xuống. Nó có thể kết nối đến 80 GPIO đến 69 ngắt ngoài để bắt trigger ngoại ( giống như ngắt ).  
*Clock system*: bao gồm các nguồn clock các nhau trong vi điều khiển như HIS, HSE, PLL, LSI RC và LSE.  
*HSI* là tín hiệu clock nội tốc độ cao có 8 Mhz RC nội. Thường được ứng dụng như clock hệ thống cho MCU mặc định.  
*HSE* là tín hiệu clock ngoại tốc độ cao và nó có thể sử dụng 4 -6 MHz xung ngoại để cung cấp tín hiệu clock ổn định để cho clock hệ thống.  
*PLL* có thể gen tín hiệu ra ổn định và chính xác từ một nguồn tần số thấp cố định. Nó kết hợp xung ra từ HIS RC hay clock thạch anh đầu ra HSE để tạo ra những tần số khác nhau cho vi điều khiển và các ngoại vi được dùng.  
*Clock LSI* là tín hiệu clock nội tốc độ thấp và tần số clock cả nó khoảng 40 KHz. Clock LSI RC có thể được dùng cho nguồn clock thấp và nó rất hữu dụng trong chế độ nghỉ hoặc dừng.  
*Dao động LSE* là thạch anh ngoại tốc độ thấp, tần số của nó vào khoảng 32 KHz và dùng cho clock thời gian thực. Nguồn clock này cấp tần số clock cho mỗi phần của vi điều khiển như CPU, các ngoại vi …   
*Startup clock*là clock hệ thống cho vi điều khiển trong trạng thái khởi động. clock này tạo ra bởi HSI RC mặc định và clock này lf khoảng 8 MHz.  
*Boot modes*xác định làm sao để boot CPU để làm việc. Có tất cả 3 chế độ boot cho vi điều khiển. Map bộ nhớ của vi điều khiển được chỉ ra dưới đây:   
Có thể sử dụng các phần khác nhau của vi điều khiển để boot CPU   
  
Hình 4.5 STM32F103 Memory Map

Một trong các chế độ boot này được giới thiệu ở dưới có thể được dùng trong Startup để boot CPU,  
User Flash: CPU sẽ boot từ Flash  
Bộ nhớ hệ thống: CPU sẽ boot từ bộ nhớ hệ thống  
SRAM: CPU sẽ boot từ SRAM.  
  
Nên chương trình bootloader là rất cần để có thể chạy và khởi chạy chương trình trên Flash bằng cách dùng USART. Bootloader được để ở bộ nhớ hệ thống.  
*Power Supply*là để cấp cho vi điều khiển để từ đó cấp cho các phần khác nhau của hệ thống. Nó được mô tả chân phải kết nối đến bộ nguồn:   
Vdd: Pin cấp nguồn cho IO và ổn áp. Vdd nên được nối đến nguồn 2.0 đến 3.6V và điện áp này cấp qua chân Vdd trên MCU.  
Vssa,Vdda: chân này cấp nguồn 1 chiều ngoài cho ADC, khối Reset, RCs và PLL. Các chân này nên được nối đến điện áp 2.0 đến 3.6 V   
Vi điều khiển này bao gồm POR/PDR, đây là một mạch tích hợp và nó thường được kích hoạt để luôn giữ 2.0 V. Khi Vdd nhỏ hơn 1 ngưỡng nào đó, vi điều khiển không thể hoạt động và nó sẽ giữ ở trạng thái reset. PVD so sánh Vdd với ngưỡng Vpvd. ISR báo đến vi điều khiển hoặc đưa nó vào trạng thái an toàn, nếu Vdd lên hoặc xuống so với Vpvd  
*RTC*và thanh ghi Backup cấp một bộ đếm chương trình 32 bit cho MCU để cấp đặt bộ đếm hoạt động liên tục. Nó được dùng để cấp cho hàm lập lịch, ngắt chu kì và ngắt cảnh báo. Nguồn được cấp bởi Vdd hay Vbat cho RTC và thanh ghi Backup.  
*General Purpose Timers (TIMx)*tạo ra 3 timer chuẩn có thể đồng bộ để dùng cho MCU. Mỗi một timer bao gồm bộ đếm counter 16 bit, bộ băm xung.  
*Advanced Control Timer (TIM1)*giống với TIMx nếu nó được cấu hình như timer 16 bit chuẩn, nhưng được hoàn thiện hơn. Nó có thể dùng 4 kênh cho Input capture, Output compare, PWM… Nó cũng có thể làm việc với 3 pha băm xung kết hợp trên 6 kênh.  
*I2C*là giao diện bus và có thể đưa ra trong 2 chế độ là master và slave. Vi điều khiển này chưa lên đên 2 bus I2C.   
GPIO là giao diện của MCU để kết nối đến các thiết bị bên ngoài có thể sử dụng như là đầu vào để đọc dữ liệu và dùng như đầu ra để ghi dữ liệu… Trên đây có sẵ 3 GPIO trên MCU ( STM32F103 với 80 chân. Nó nên được cấu hình mềm và được cân nhắc trong suốt lúc làm việc với MCU. Tất cả các ngoại vi của MCU có thể kết nối ra bên ngoài hoặc các thiết bị ngoài khác bở GPIO

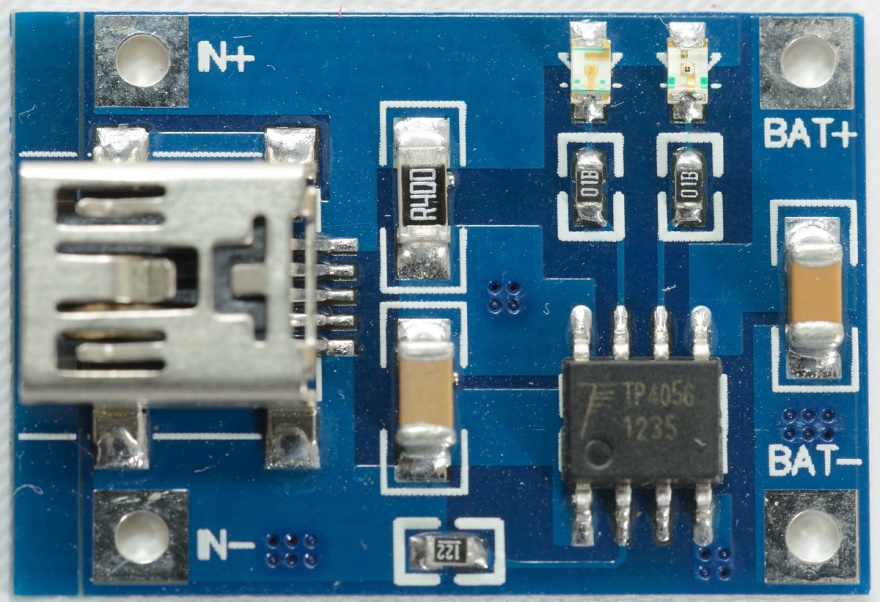
**Cấu hình vi điều khiển**Cấu hình vi điều khiển đặt xung clock cho CPU của vi điều khiển và các ngoại vi của nó. Cấu hình cho kiểu bộ dao động của vi điều khiển và đặt chính các tần số cho mỗi một phần của vi điều khiển. Nó cũng xác định chân vi điều khiển cho các ngoại vi như là I/O, chế độ hoạt động … . Nó cấu hình những ngoại vi cần thiết và được chọn ra trong dự án. Với đề tài này thì chúng ta chọn GPIO.

**Cấu hình RCC**Bước đầu tiên để cấu hình là chuẩn bị clock hệ thống cho vi điều khiển và nó rất quan trong trong lần đầu tiên vì clock hệ thống bật lên trên vi xử lý và việc kích hoạt ngoại vi cũng là rất cần thiết. Bằng việc cấu hình RCC, nó có thể chạy clock hệ thống.   
HSE được xác định như clock nguồn cho vi điều khiển, bởi vì nó có thể hoạt động ở 72 MHz (tần số clock hệ thống cực đại) nếu HSE được dùng như clock PLL đầu vào. Một lý do khác là nó có thể đưa ra tần số clock cho MCU rất chính xác. Vậy nên, HSE được chuẩn bị để đưa ra như tần số clock nguồn cho CPU và mỗi ngoại vi trên vi điều khiển.  
  
  
Hình 4.6 Clock tree for microcontroller STM32F103xx  
  
Sau khi cấu hình, tần số clock của CPU và các ngoại vi có thể được đặt. Sau đó CPU có thể bắt đầu làm việc và các ngoại vi thì sẵn sàng sử dụng.  
Hình trên cũng đưa ra máy dao động HSE được dùng như clock đầu vào PLL và đầu ra của PLL là nguồn clock hệ thống. Tần số SYSCLK nên thay đổi, căn chỉnh theo mỗi phần khác nhau của vi điều khiển, bởi vì mỗi ngoại vi làm việc với những tần số clock riêng biệt.

**Cấu hình GPIO**  
Cấu hình GPIO chọn ra và kích hoạt các port của vi điều khiển hay bất kì chân nào của vi điều khiển để dùng cho vi điều khiền trong các task. Cấu hình này xác định IO và trạng thái của mỗi port hay mỗi chân.   
Hình 4.7 Mô tả cấu hình chân của vi điều khiển STM32F103. Việc cấu hình cho các chân này là rất quan trong để cấy hình các port hay chân của vi điều khiển cho chương trình và nó nên được cân nhắc. Số lượng chân và tính chất của nó được nhắc đến ở hình dưới và nó rất hữu ích cho việc cấu hình các chân của vi điều khiển.  
  
  
Hình 4.7 Pin Configuration of the STM32F103  
  
Các ngoại vi của vi điều khiển có sẵn từ các chân của Port A, B, C. Vậy nên, việc kích hoạt các port này bởi chương trình là rất cần thiết theo yêu cầu công việc và nó có được ở việc cấu hình GPIO. Trong cấu hình, clock cho GPIO được kích hoạt cho tất cả các port để kích hoạt port của nó.

**Cấu hình NVIC**Trong việc cấu hình NVIC, ưu tiên ngắt của các ngoại vi được đặt theo chương trình.

* 1. **Module sạc pin có bảo vệ dùng TP4056**

****Hình 4.8 Module sạc pin dùng TP4056

* + 1. **Tính năng**

- Sạc Pin 3.7V : Pin Ultrafire, Pin Lion 3.7V  
- Báo Sạc Đầy.  
- Chức năng bảo vệ xả cạn pin.

* + 1. **Thông số kỹ thuật**

- Điện áp vào: DC 5V  
- Cổng Sạc MicroUSB  
- Điện Áp Sạc: 4.2V  
- Dòng sạc tối đa: 1A  
- Bảo vệ xả đến điện áp: 2.5V ( Tránh làm cạn Pin)  
- Bảo Vệ Xả Quá Dòng: 3A  
- Kích thước: 26x17MM

* 1. **Module tăng áp**

Hình 4.9 Module tăng áp

* + 1. **Tính năng**

Tăng áp từ điện áp dưới 5V DC lên đến 5V DC

* + 1. **Thông số kỹ thuật**

**-** Thuộc dòng module boost   
- Điện áp vào: 1-5V  
- Điện áp đầu ra: 5,1-5,2V   
- Hiệu suất chuyển đổi 96 %   
- Dòng ra 0.5A   
- Nhiệt độ làm việc: -40 – 85 độ C.  
- Không tự bảo vệ khi ngắn mạch.

* 1. **Màn hình LCD01006A**

****Hình 4.10 LCD01006A

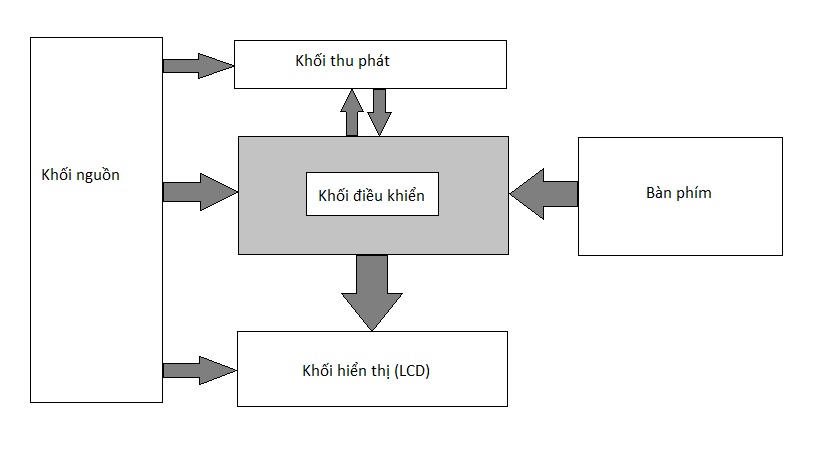
* + 1. **Tính năng**

**LCD 8x2** có thể hiện thị được 2 dòng mỗi dòng 8 ký tự , màn hình có thể điều chỉnh được tương phản có tích hợp đèn nền xanh dương

* + 1. **Thông số kỹ thuật**

**-** Điện áp hoạt động: 5V DC  
- Kích thước : 58.0 x 32.0 x 14 mm  
- Cỡ chữ: 2.48 x 5.00 mm   
- Cỡ chấm 0.45 x 0.55 mm   
- Driver: S6A0069

1. **Sơ đồ khối và nguyên lý**
   1. **Sơ đồ khối**

  
Hình 4.11 Sơ đồ khối của mạch điều khiển từ xa bằng hồng ngoại  
  
Mạch gồm có 5 khối cơ bản:   
- Khối điều khiển: có nhiệm vụ điều khiển hoạt động cả mạch (sử dụng vi điều khiển stm32f103)  
-Khối thu phát: (thu phát hồng ngoại) để thu tín hiệu học lệnh và phát tín hiệu điều khiển  
- Khối hiển thị (LCD): hiện thị thông tin lên màn hình ( thông tin mã lệnh học được và mã lệnh gửi đi )  
- Bàn phím: Nhận lệnh điều khiển từ người dùng (gồm các phím chức năng, reset, mode, và Get IR code )  
- Khối nguồn: cấp nguồn hoạt động cho cả mạch.( cáp nguồn hoạt động cho các khối còn lại trong mạch)

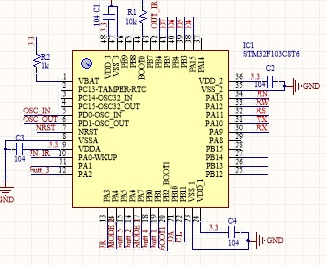
* 1. **Nguyên lý hoạt động**
     1. **Chế độ học lệnh**

Ở chế độ này, mạch đưa sang trạng thái sẵn sàng nhận mã IR từ nguồn hồng ngoại bên ngoài. Vi điều khiển set cho IR ở chế độ đọc dữ liệu (). Khối điều hiển nhận lệnh học mã từ button của mạch nút nhấn. Khi đưa thiết bị phát lệnh hồng ngoại vào đầu thu hồng ngoại của mạch, khối điều khiển nhận mã lệnh thu được từ hồng ngoại lưu vào bộ nhớ. Ứng với mỗi nút nhấn là 1 vùng nhớ của mã lệnh. Sau khi nhận mã lệnh từ thiết bị phát hồng ngoại và lưu vào bộ nhớ, khối điều khiển đợi cho chế độ học lệnh được nhấn trở lại (switch mode) set IR về chế độ phát mã lệnh hồng ngoại.

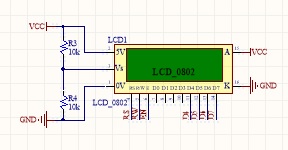
* + 1. **Chế độ điều khiển**

Với chế độ điều khiển, mã lệnh đã được lưu vào bộ nhớ được sử dụng cho các button ở mạch nút nhấn. Mạch đợi nhận tín hiện vào từ mạch nút nhấn , ứng với mỗi nút nhấn là 1 mã lệnh, ngắt xảy ra, đọc button để đưa ra mã lệnh phát đi qua hồng ngoại. Sau khi phát mã lệnh thành công, mạch lại được đưa về chế độ đợi phím, cờ ngắt reset về 0.

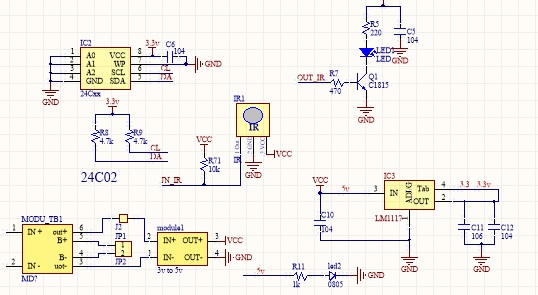
1. **Thiết kế mạch nguyên lý và mạch in**
   1. **Mạch nguyên lý**

****

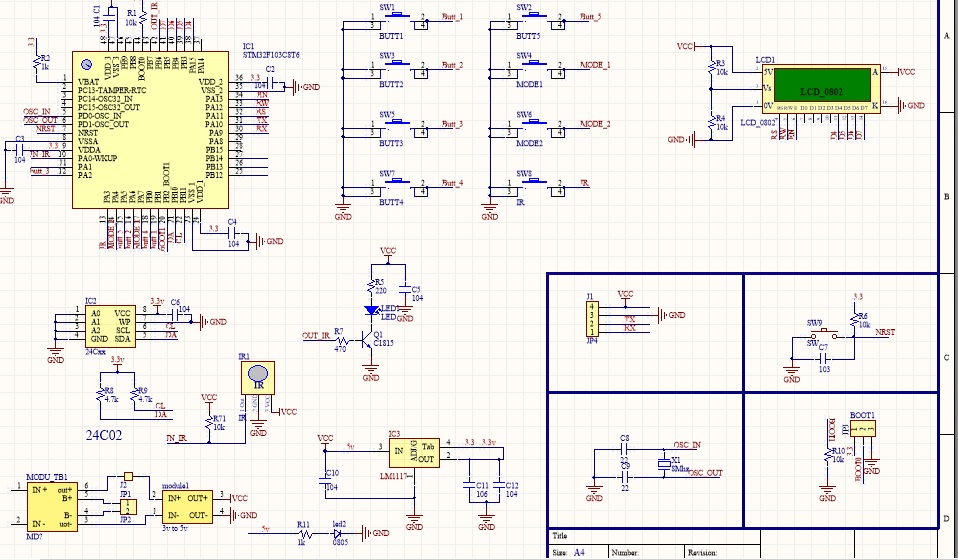
Hình 4.12 Khối điều khiển



Khối hiển thị

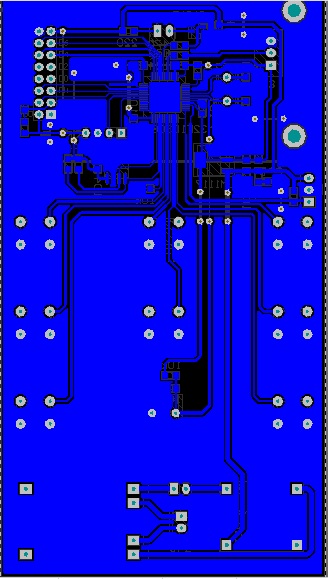


Hình 4.13 Khối thu phát tín hiệu và mạch nguồn

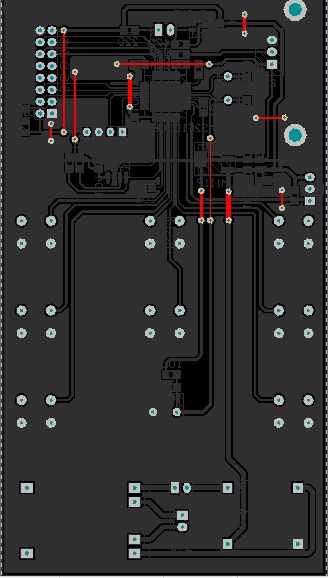


Hình 4.14 Mạch nguyên lý

* 1. **Mạch in**

****

Hình 4.15 Mạch in (bottom)

Hình 4.16 Mạch in (top)

1. **Thuật toán xử lý và lập trình** 
   1. **Thuật toán**
      1. **Lưu đồ thuật toán**

****

* + 1. **Giải thích thuật toán**

Mạch ban đầu được đưa về trạng thái nhận nút bấm. Sau khi nhận được dữ liệu từ nút nhấn, vi điều khiển kiểm tra chế độ nhận lệnh hay phát lệnh của mạch. Nếu là nhận lệnh, vi điều khiển kiểm tra MODE, tương ứng với 2 thiết bị muốn điều khiển. Nếu là thiết bị 1, Vi điều khiển nhận lệnh từ các nút nhấn chức năng của mạch gồm 5 nút. Sau khi chọn được phím chức năng cần lấy IR code thì mạch đưa về trạng thái chờ dữ liệu từ đèn thu hồng ngoại. Nhận được đủ mã lệnh, vi điều khiển hiển thị mã lệnh nhận được ra màn hình LCD đồng thời lưu mã lệnh vào EEPROM. Kết thúc mạch lại đưa về trạng thái đợi nút nhấn. Tương tự , khi nhấn chuyển sang thiết bị 2, mạch cũng thu dữ liệu hồng ngoại tương ứng với các nút nhấn trong mạch, hiển thi và lưu vào bộ nhớ.  
Với chế độ phát lệnh, mạch cũng kiểm tra MODE tương ứng với thiếu bị 1 hay thiết bi 2. Nếu là thiết bị 1, vi điều khiển in ra màn hình MODE 1 cùng với mã IR học được lấy từ trong EEFROM, đông thời đẩy dữ liệu đi qua đèn phát hồng ngoại. Tương ứng với mỗi phím chức năng đã được setup thì cũng tương ứng là các mã IR phát đi. Sau khi quá trình phát mã lệnh thành công, mạch đưa về trạng thái ban đầu là nhận nút nhấn BUTTON(). Tương tự với thiết bị 2, vi điều khiển cũng hiển thi lên màn hình LCD MODE2 và mã lệnh học được tương ứng ở mỗi nút và cuối cùng là đưa mạch về trạng thái ban đầu BUTTON().

* 1. **Lập trình** 
     1. **Hàm ghi dữ liệu**

|  |
| --- |
| uint8\_t Write\_data(uint8\_t byte,uint8\_t \* data)  {  uint8\_t byte\_data;  if (byte==0) byte\_data = BIT\_BYTE(data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7]);  else if(byte==1) byte\_data = BIT\_BYTE(data[8],data[9],data[10],data[11],data[12],data[13],data[14],data[15]);  else if(byte==2) byte\_data = BIT\_BYTE(data[16],data[17],data[18],data[19],data[20],data[21],data[22],data[23]);  else if(byte==3) byte\_data = BIT\_BYTE(data[24],data[25],data[26],data[27],data[28],data[29],data[30],data[31]);  return byte\_data;  } |

* + 1. **Hàm đưa dữ liệu lên màn hình**

|  |
| --- |
| void View\_data(uint8\_t \* data)  {  uint32\_t DATA\_IR;  DATA\_IR=data[0];  DATA\_IR=DATA\_IR<<8;  DATA\_IR=DATA\_IR|data[1];  DATA\_IR=DATA\_IR<<8;  DATA\_IR=DATA\_IR|data[2];  DATA\_IR=DATA\_IR<<8;  DATA\_IR=DATA\_IR|data[3];  target=0;  for(i=0;i<4;i++)  {  printf("%X|",data[i]);  }  printf("\r\n");  target=1;  LCD\_Gotoxy(0,1);  printf("%8.X ",DATA\_IR);  } |

* + 1. **Hàm GetByte**

|  |
| --- |
| uint8\_t BIT\_BYTE(uint8\_t D0, uint8\_t D1, uint8\_t D2, uint8\_t D3, uint8\_t D4, uint8\_t D5, uint8\_t D6, uint8\_t D7)  {  uint8\_t data;  data= D7\*128 + D6\*64 + D5\*32 + D4\*16 + D3\*8 + D2\*4  + D1\*2 +D0\*1; // 0 den 255  return data;  } |

* + 1. **NOT\_BIT\_BYTE**

|  |
| --- |
| uint8\_t NOT\_BIT\_BYTE(uint8\_t D0, uint8\_t D1, uint8\_t D2, uint8\_t D3, uint8\_t D4, uint8\_t D5, uint8\_t D6, uint8\_t D7)  {  uint8\_t data;  data= (!D7)\*128 + (!D6)\*64 + (!D5)\*32 + (!D4)\*16 + (!D3)\*8 + (!D2)\*4 + (!D1)\*2 +(!D0)\*1; // 0 den 255  return data;  } |

* + 1. **Hàm cấu hình ngắt**

|  |
| --- |
| void ext\_line0\_init(void)  {  NVIC\_InitTypeDef NVIC\_InitStructure;  EXTI\_InitTypeDef EXTI\_InitStructure;  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;  GPIO\_Init(GPIOA, &GPIO\_InitStructure);  GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource0);  EXTI\_ClearITPendingBit(KEY\_BUTTON\_EXTI\_LINE0);  EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;  EXTI\_InitStructure.EXTI\_Line = KEY\_BUTTON\_EXTI\_LINE0;  EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;  EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;  EXTI\_Init(&EXTI\_InitStructure);  NVIC\_InitStructure.NVIC\_IRQChannel = EXTI0\_IRQn;  NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 15;  NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;  NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;  NVIC\_Init(&NVIC\_InitStructure);  } |

Hàm cấu hình ngắt cho line 0 , Đặt vào chân A0 của vi điều khiển chế độ input. Cấu hình cho ngắt sườn lên, Đặt priority thấp nhất 15

* + 1. **Hàm cấu hình IO**

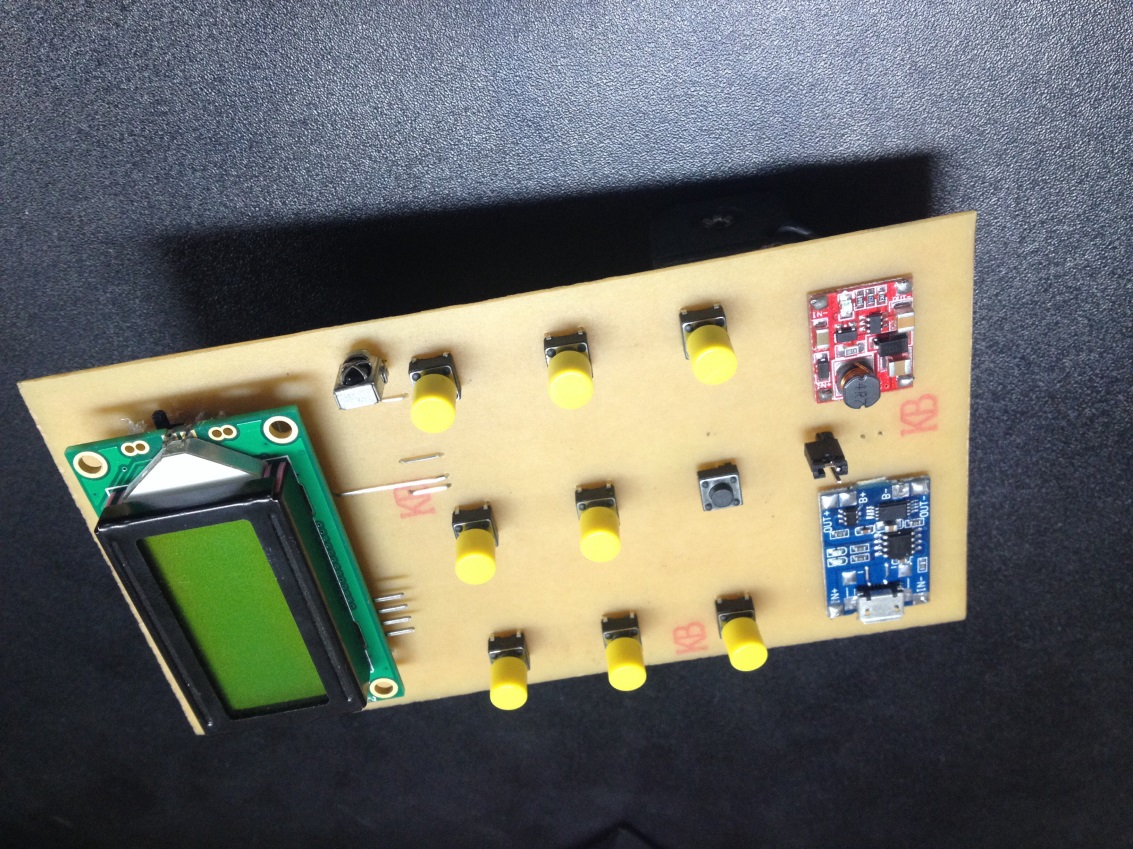
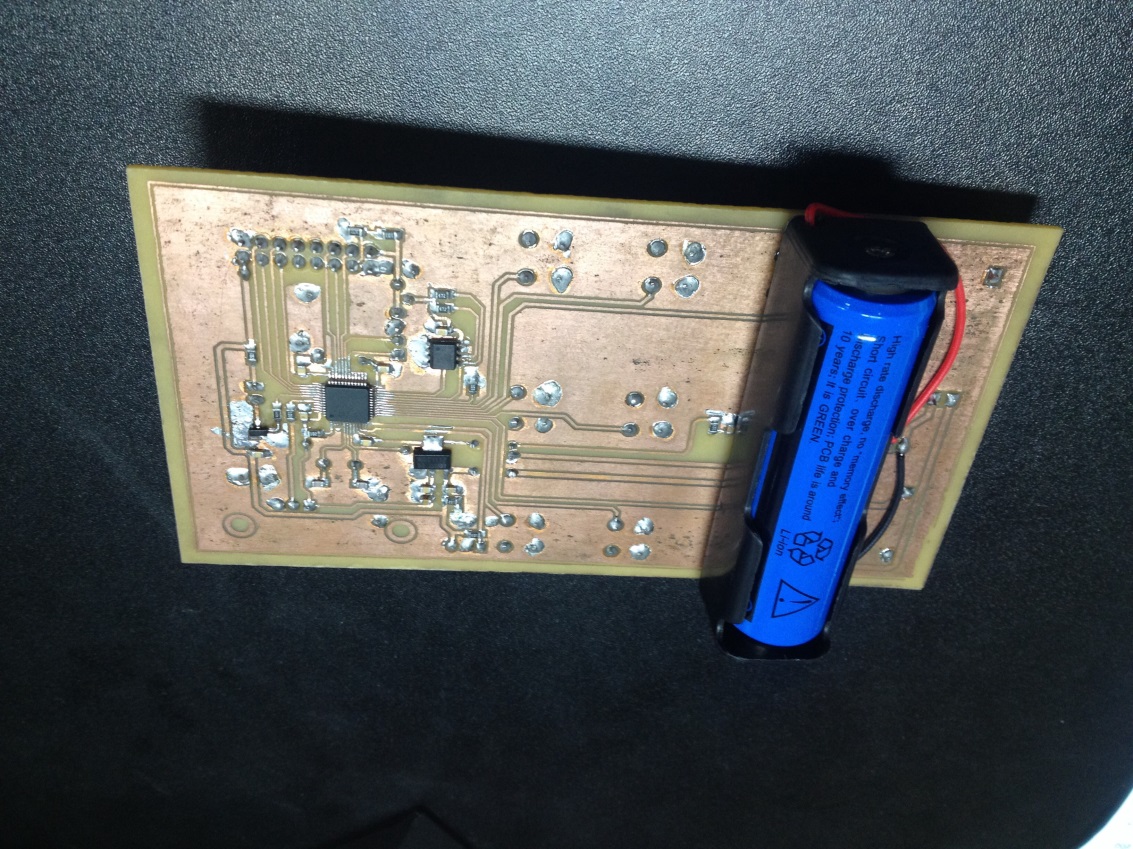
|  |
| --- |
| RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA; RCC\_APB2Periph\_GPIOB |RCC\_APB2Periph\_GPIOC| RCC\_APB2Periph\_AFIO, ENABLE); GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_Disable, ENABLE); GPIO\_InitStructure.GPIO\_Pin = (GPIO\_Pin\_0|GPIO\_Pin\_1);////////butt  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;  GPIO\_Init(GPIOB, &GPIO\_InitStructure);  GPIO\_InitStructure.GPIO\_Pin = (GPIO\_Pin\_2|GPIO\_Pin\_3|GPIO\_Pin\_4|GPIO\_Pin\_5|GPIO\_Pin\_6|GPIO\_Pin\_7);//butt  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;  GPIO\_Init(GPIOA, &GPIO\_InitStructure); GPIO\_InitStructure.GPIO\_Pin = (GPIO\_Pin\_1); GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; GPIO\_Init(GPIOA, &GPIO\_InitStructure); |

* + 1. **Hàm cấu hình UART**

|  |
| --- |
| void USART1\_Configuration(void)  {  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;  GPIO\_Init(GPIOA, &GPIO\_InitStructure);  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;  GPIO\_Init(GPIOA, &GPIO\_InitStructure);  USART\_InitStructure.USART\_BaudRate = 115200;  USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;  USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;  USART\_InitStructure.USART\_Parity = USART\_Parity\_No;  USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;  USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;  USART\_Init(USART1, &USART\_InitStructure);  USART\_Cmd(USART1, ENABLE);  } |

Hàm cấu hình UART1 đặt vào chân A9 và A10 tương đương là 2 chân Tx và Rx. A9 chế độ push-pull ở speed 50Mhz, A10 ở mode floating.  
BaudRate 11520, length 8 bit, 1 bit stop, không có priority, RTS và CTS bị disable, ứng với 2 chân Tx, Rx được config là transmit và receive.  
Sau khi cấu hình hoàn tất, enable USART.

1. **Kết quả**

Mặt trước:   
****  
Mặt sau   
****  
  
Mạch chạy ổn định với các thiết bị sử dụng chuẩn NEC. Quá trình học lệnh của các phím chức năng nhanh, chính xác và thao tác đơn giản. Quá trình phát lệnh điều khiển của các phím ổn định, không gặp lỗi nhờ vào ưu điểm của giao thức NEC.   
Hạn chế: Mạch chỉ sử dụng được với các thiết bị dùng chuẩn NEC vì không thể biết được tần số sóng mang của các chuẩn khác, có hạn chế về số phím chức năng trên mạch, số thiết bị điều khiển chỉ giới hạn là 2 thiết bị.

# Kết luận

*Kết quả đạt được:* Đề tài tạo ra mạch điều khiển từ xa học lệnh qua hồng ngoại đạt được những kết quả tích cưc. Mạch điều khiển có thể học lệnh từ các điều khiển khác dùng NEC với tỉ lệ lỗi thấp, quá trình học lệnh nhanh, thao tác đơn giản. Mã được hiện thị ra màn hình gần như ngay sau khi nguồn phát tín hiệu đi, cho thấy tốc độ xử lý vượt trội của vi điều khiển STM32F103, tỉ lệ truyền lỗi thấp là ưu điểm của chuẩn NEC khi các mã lệnh được truyền đi 2 lần, với lần 2 là đảo bit của lần 1. Bộ nhớ EEFROM tích hợp trong mạch lưu được số lượng mã lệnh lớn, khả năng mở rộng thiết bị vì thế được nâng cao. Quá trình truyền mã lệnh tốc độ cao, chính xác khi thiết bị điện phản hồi tức thì. Phạm vi điều khiển, nhận lệnh cao lên đến 15m đáp ứng mọi nhu cầu cho các thiết bị trong 1 căn phòng. Mạch được kiểm tra tình trạng hoạt động trong thời gian dài ( onscreen 4 ngày liên tục ) vẫn hoạt động ổn định cho tới khi pin cạn.   
*Nhược điểm:* Bên cạnh đó, mạch điều khiển vẫn có những nhược điểm như chỉ điều khiển được các thiết bị dùng chuẩn NEC, số lượng thiết bị có thể điều khiển cao nhất là 2 thiết bị, kích thước mạch chưa nhỏ gọn và độ bền chưa được như kỳ vọng.   
*Hướng phát triển của sản phẩm*: Có thể lập trình để tăng số thiết bị có thể điều khiển được trong gia đình, thiết kế của mạch nhỏ gọn, trọng lượng giảm đi khi sử dụng nguồn năng lượng khác, thiết bị có thể kết nối internet thông qua wifi để cập nhật dữ liệu điều khiển của các thiết bị không dùng chuẩn NEC. Nếu thương mại hóa, sản phẩm sẽ được tối ưu về thuật toán, thiết kế các linh kiện chất lượng cao hơn, nhẹ hơn, nhỏ gọn hơn và cho thời gian sử dụng lâu hơn nữa. Dần thay thế phím bấm bằng bàn phím cảm ứng.

# Tham khảo