**Trường Đại Học Bách Khoa Hà Nội**

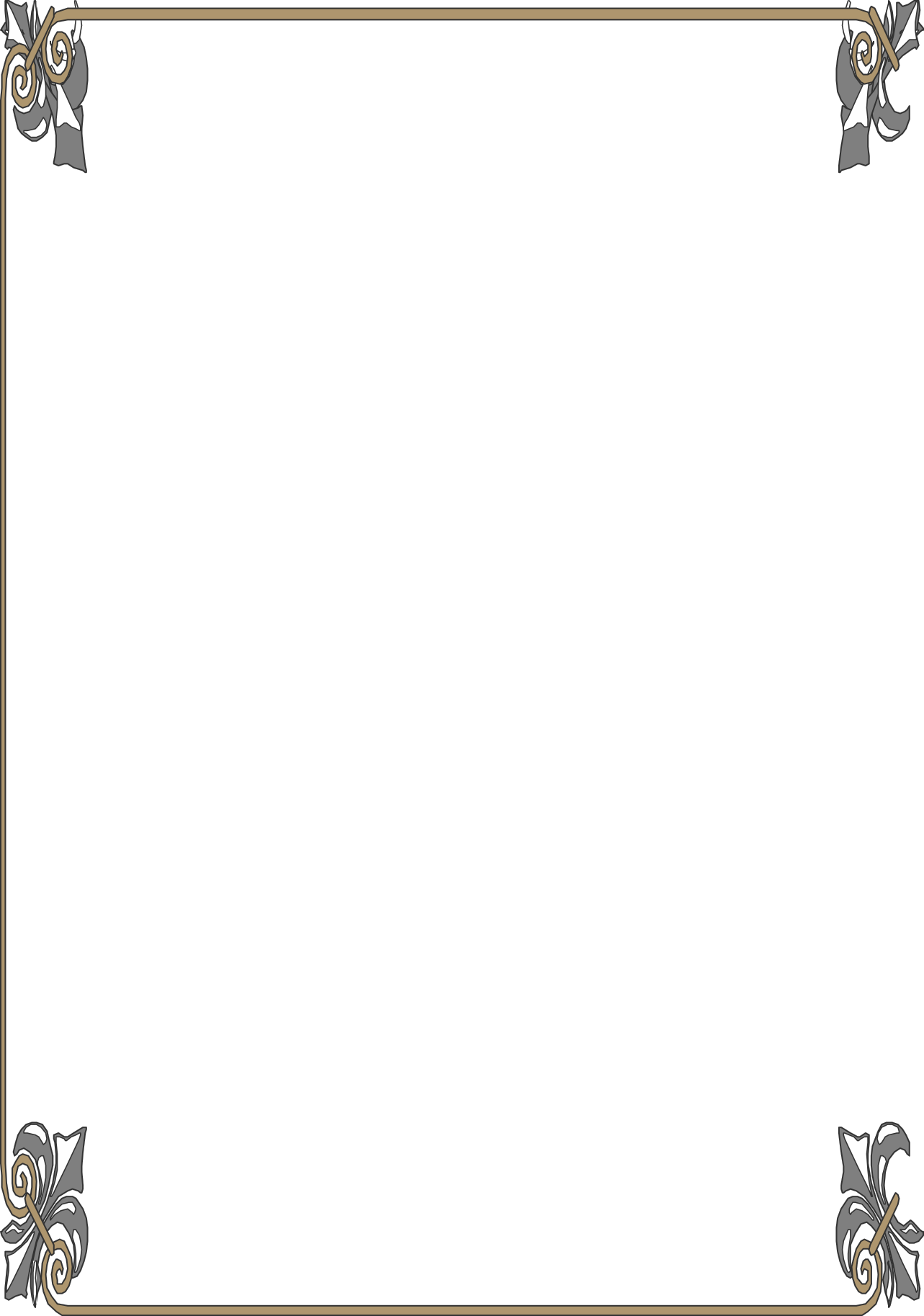
**Viện Điện Tử Viễn Thông**

..…oOo…..

**Đề tài:**

**Đồ Án Tốt Nghiệp**

**Lập trình thiết bị thu âm và phát nhạc với STM32F4**



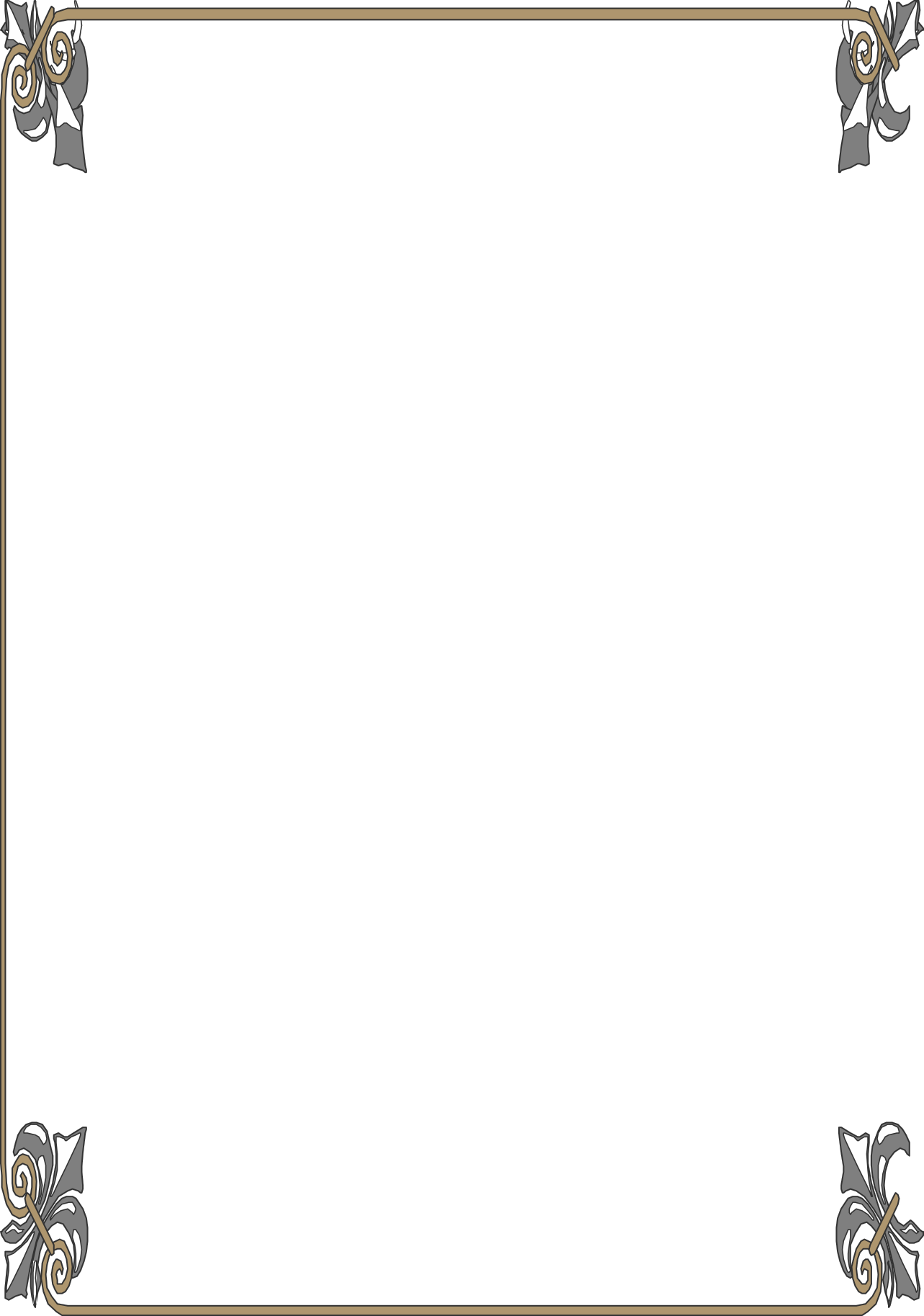
**Sinh Viên : Trần Văn Hà**

**MSSV : 20141325**

**Lớp : ĐT 04-K59**

**GVHD : TS. Lê Dũng**

**Ha Noi, 06/2019**



**Trường Đại Học Bách Khoa Hà Nội**

**Viện Điện Tử Viễn Thông**

..…oOo…..

**Đề tài:**

**Đồ Án Tốt Nghiệp**

**Lập trình thiết bị thu âm và phát nhạc với STM32F4**

**Sinh Viên : Trần Văn Hà**

**MSSV : 20141325**

**Lớp : ĐT 04-K59**

**GVHD : TS. Lê Dũng**

**Giảng Viên phản biện:……………………………**

**Ha Noi, 06/2019**

**Đánh giá quyển đồ án tốt nghiệp (Dùng cho giảng viên hướng dẫn)**

Giảng viên đánh giá:……...**Dr. Lê Dũng**………………………. Họ và tên Sinh viên:.……..**Trần Văn Hà**……………………… MSSV:……..……………..**20141325**…………………………… Tên đồ án: **Lập trình thiết bị ghi âm và phát nhạc với STM32**

***Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Có sự kết hợp giữa lý thuyết và thực hành (20)** | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả  thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án | 1 | 2 | 3 | 4 | 5 |
| 2 | Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế) | 1 | 2 | 3 | 4 | 5 |
| 3 | Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề | 1 | 2 | 3 | 4 | 5 |
| 4 | Có kết quả mô phỏng/thưc nghiệm và trình bày rõ ràng kết quả đạt được | 1 | 2 | 3 | 4 | 5 |
| **Có khả năng phân tích và đánh giá kết quả (15)** | | | | | | |
| 5 | Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống | 1 | 2 | 3 | 4 | 5 |
| 6 | Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng. | 1 | 2 | 3 | 4 | 5 |
| 7 | Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận  để đề xuất hướng giải quyết có thể thực hiện trong tương lai. | 1 | 2 | 3 | 4 | 5 |
| **Kỹ năng viết (10)** | | | | | | |
| 8 | Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận  chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | 1 | 2 | 3 | 4 | 5 |
| 9 | Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.) | 1 | 2 | 3 | 4 | 5 |
| **Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)** | | | | | | |
| 10a | Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh  sáng chế | | 5 | | | |
| 10b | Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về  chuyên ngành như TI contest. | | 2 | | | |
| 10c | Không có thành tích về nghiên cứu khoa học | | 0 | | | |
| **Điểm tổng** | | | **/50** | | | |
| **Điểm tổng quy đổi về thang 10** | | |  | | | |

***3. Nhận xét thêm của Thầy/Cô (****giảng viên hướng dẫn nhận xét về thái độ và tinh thần làm việc của sinh viên****)***

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

Ngày /06/2019 Người nhận xét

(Ký và ghi rõ họ tên)

**Đánh giá quyển đồ án tốt nghiệp**

**(Dùng cho cán bộ phản biện)**

Giảng viên đánh giá:……………………………………………………………… Họ và tên Sinh viên**:**……….**Trần Văn Hà**…………………................................ MSSV:……………………..**20141325**…………..………………………………. Tên đồ án: **lập trình thiết bị ghi âm và chơi nhac với STM32**

***Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây: Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Có sự kết hợp giữa lý thuyết và thực hành (20)** | | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng  dụng của đồ án | | 1 | 2 | 3 | 4 | 5 |
| 2 | Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế) | | 1 | 2 | 3 | 4 | 5 |
| 3 | Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề | | 1 | 2 | 3 | 4 | 5 |
| 4 | Có kết quả mô phỏng/thưc nghiệm và trình bày rõ ràng kết quả đạt được | | 1 | 2 | 3 | 4 | 5 |
| **Có khả năng phân tích và đánh giá kết quả (15)** | | | | | | | |
| 5 | Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực  hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống | | 1 | 2 | 3 | 4 | 5 |
| 6 | Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng. | | 1 | 2 | 3 | 4 | 5 |
| 7 | Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. | | 1 | 2 | 3 | 4 | 5 |
| **Kỹ năng viết (10)** | | | | | | | |
| 8 | Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận  chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | | 1 | 2 | 3 | 4 | 5 |
| 9 | Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.) | | 1 | 2 | 3 | 4 | 5 |
| **Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)** | | | | | | | |
| 10a | | Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế | 5 | | | | |
| 10b | | Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về  chuyên ngành như TI contest. | 2 | | | | |
| 10c | | Không có thành tích về nghiên cứu khoa học | 0 | | | | |
| **Điểm tổng** | | | **/50** | | | | |
| **Điểm tổng quy đổi về thang 10** | | |  | | | | |

***3. Nhận xét thêm của Thầy/Cô***

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

.......................................................................................................................................................

Ngày …/06/2019 Người nhận xét (Ký, ghi rõ họ tên)

# Lời nói đầu

Trước tiên, em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên hướng dẫn của mình, Tiến sĩ Lê Dũng, Bộ môn Điện Tử và Kỹ Thuật Máy Tính, Viện Điện Tử Viễn Thông, Trường Đại Học Bách Khoa Hà Nội. Trong suốt thời gian em thực hiện đồ án, Thầy đã tận tình hướng dẫn, đưa ra hướng phát triển cũng như cách giải quyết các khó khắn để em hoàn thành đồ án đúng hạn.

Tôi cũng xin bày tỏ long biết ơn đến tập thể anh em trong sự án SDK, FGA.DPA, tập đoàn FPT. Trong thời gian tôi làm việc tại đây, cũng chính là thời gian tôi làm đồ án tốt nghiệp, các anh chị Project manager, Technical Lead đã cung cấp cho tôi các điều kiện về cơ sở vật chất cũng linh kiện, thiết bị. Các anh em developer đã luôn tận tình giúp đỡ, chỉ bảo cho tôi không chỉ những kiến thức về lập trình, kiến trúc vi điều khiển mà còn cả những kỹ năng làm việc nhóm, kỹ năng tiếng anh, ký năng giao tiếp với khách hang hay hiểu biết về quy trình phát triển phần mềm. Đó thực sự là những kiến thức và trải nghiệm rất quý giá mà nhà trường không bao giờ dạy. Những điều đó giúp tôi tự tin hơn rất nhiều và vững bước trên con đường tôi đã chọn, con đường trở thành một kỹ sư lập trình nhúng.

Sau cùng, Con xin bày tỏ lòng biết ơn sâu sắc đến bố, mẹ. Những người luôn đứng sau, ủng hộ con vô điều kiện và là động lực giúp con vượt qua những khó khăn. Không có gia đình phía sau, con không thể có được những thành quả như ngày hôm nay.

Ha Noi, ngày 10 tháng 06 năm 2019

Ha Tran Van

**Lời Cam Đoan**

Tôi là Trần Văn Hà, mã số sinh viên 20141325, sinh viên lớp DT 04, khóa 59. Người hướng dẫn là TS. Lê Dũng. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án *lập trình thiết bị thu âm và phát nhạc với STM32F4* là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

|  |  |
| --- | --- |
|  | Hà nội, ngày 10 tháng 06 năm 2019  **Người cam đoan**  **Trần Văn Ha** |

# Mục Lục

[ACKNOWLEDGEMENTS 7](#_bookmark0)

[Commitment 8](#_bookmark1)

[Contents 9](#_bookmark2)

[Acronyms 11](#_bookmark3)

[List of Figure 12](#_bookmark4)

[List of Table 14](#_bookmark5)

[ABSTRACT 15](#_bookmark6)

[CHAPTER I: PROJECT SUMMARY 16](#_bookmark7)

* 1. [Project summary 16](#_bookmark8)
  2. [Aims of research 16](#_bookmark9)
  3. [Research methods 17](#_bookmark10)
  4. [Conclusion 17](#_bookmark11)

[CHAPTER II: UNDERSTANDING THE THEORY 18](#_bookmark12)

* 1. [FAT File system 19](#_bookmark13)
     1. [History of FAT File system 19](#_bookmark14)
     2. [Technical overview 19](#_bookmark15)
     3. [FAT File system layout 20](#_bookmark16)
  2. [Digital to Analog Converter 22](#_bookmark18)
     1. [DAC peripheral on STM32F407VG MCU 23](#_bookmark19)
     2. [DAC functional description 24](#_bookmark21)
     3. [DAC operation summary 26](#_bookmark26)
  3. [Direct Memory Access (DMA) 26](#_bookmark27)
     1. [DMA principles 27](#_bookmark28)
     2. [Bus mastering 27](#_bookmark29)
     3. [Transfer types 27](#_bookmark30)
     4. [DMA operation modes 28](#_bookmark31)
     5. [DMA module intergraded in STM32F407VG 29](#_bookmark32)
     6. [DMA functional description 31](#_bookmark33)
  4. [Serial Peripheral Interface/ Inter-IC Sound (SPI/I2S) 44](#_bookmark43)
     1. [SPI/I2S Introduction 44](#_bookmark44)
     2. [SPI and I2S main features 45](#_bookmark45)
     3. [SPI functional description 47](#_bookmark46)
     4. [I2S functional description 51](#_bookmark50)
  5. [I2C protocol 55](#_bookmark53)
     1. [Introduction 55](#_bookmark54)
     2. [I2C bus features 55](#_bookmark55)
     3. [I2C mode 56](#_bookmark57)
     4. [The I2C-bus protocol 57](#_bookmark58)
  6. [USB protocol 62](#_bookmark65)
     1. [USB Communication 63](#_bookmark67)
     2. [USB Advantages and Disadvantages 63](#_bookmark69)
  7. [WAV File Format 64](#_bookmark71)
     1. [Wave Audio File Format 64](#_bookmark72)
     2. [Wave file header 65](#_bookmark74)
     3. [Format Chunk 66](#_bookmark76)
     4. [Data chunk 66](#_bookmark77)

[CHAPTER III: HARDWARE AND DEVELOPMENT TOOLS 68](#_bookmark79)

* 1. [STM32F4 Discovery kit 69](#_bookmark80)
     1. [Features 69](#_bookmark81)
     2. [Hardware structure 69](#_bookmark82)
     3. [Embedded Debug Interface 70](#_bookmark85)
     4. [On-board audio capability 71](#_bookmark88)
  2. [Keil C uVison 5 IDE 72](#_bookmark89)
  3. [STM32 CubeMx 73](#_bookmark93)

[CHAPTER IV: SYSTEM DESIGN 75](#_bookmark97)

* 1. [Application overview 76](#_bookmark98)
  2. [Audio Playback Application 77](#_bookmark101)
  3. [Audio Recorder Application 77](#_bookmark103)
  4. [Volume control 78](#_bookmark105)

[CHAPTER V: CONCLUSION AND REFERENCE 80](#_bookmark108)

[Conclusion 80](#_bookmark109)

[Reference 81](#_bookmark110)

**Danh Mục Từ Viết Tắt**

|  |  |
| --- | --- |
| **Acronyms** | **Full word** |
| DAC | Digital to analog converter |
| I2C | Inter Intergraded Circuit |
| I2S | Inter-IC sound |
| SPI | Serial Peripherals Interface |
| USB | Universal Serial Bus |
| FAT | File Allocation Table |
| DMA | Direct Memory Access |
| IDE | Intergraded Development Environments |
| ACK | Acknowledgement |
| NACK | Non-Acknowledgement |
| SDA | Serial Data |
| SCL | Serial Clock |
| MOSI | Master Out Slave In |
| MISO | Master In Slave Out |
| CS | Chip Select |
| SD | Serial Data |
| WS | Word Select |

# Danh Mục Hình Ảnh

## [Figure 1 DAC channel block diagram 24](#_bookmark20)

## [Figure 2 DAC control register bit map 25](#_bookmark22)

## [Figure 3 DAC channel 1 12-bit right-aligned data holding register 25](#_bookmark23)

## [Figure 4 DAC channel 1 data output register 25](#_bookmark24)

## [Figure 5 DAC status register 26](#_bookmark25)

## [Figure 6 DMA block diagram 31](#_bookmark34)

## [Figure 7 System implementation of the two DMA controllers on STM32F407 32](#_bookmark35)

## [Figure 8 DMA channel selection 33](#_bookmark36)

## [Figure 9 DMA1 request mapping 33](#_bookmark37)

## [Figure 10 DMA2 request mapping 34](#_bookmark38)

## [Figure 11 DMA direction bitmap 35](#_bookmark39)

## [Figure 12 Peripheral-to-memory mode 36](#_bookmark40)

## [Figure 13 Source and destination address registers in Double buffer mode 38](#_bookmark41)

## [Figure 14 DMA FIFO structure 40](#_bookmark42)

## [Figure 15 The block diagram of the SPI 47](#_bookmark47)

## [Figure 16 Single master/ single slave application 48](#_bookmark48)

## [Figure 17 Data clock timing diagram 50](#_bookmark49)

## [Figure 18 I2S block diagram 51](#_bookmark51)

## [Figure 19 I2S Philips protocol waveforms 53](#_bookmark52)

## [Figure 20 Example of I2C-bus applications 56](#_bookmark56)

## [Figure 21 Devices with various supply voltages sharing the same bus 57](#_bookmark59)

## [Figure 22 Bit transfer on the I2C-bus 58](#_bookmark60)

## [Figure 23 START and STOP conditions 58](#_bookmark61)

## [Figure 24 Data transfer on the I2C-bus 59](#_bookmark62)

## [Figure 25 A complete data transmission 60](#_bookmark63)

## [Figure 26 START byte procedure 61](#_bookmark64)

## [Figure 27 USB flash drive 62](#_bookmark66)

## [Figure 28 Logical Connections between USB Host Clients and USB Device](#_bookmark68)

## [Endpoints 63](#_bookmark68)

## [Figure 29 WAV File Format Layout 65](#_bookmark73)

## [Figure 30 Hardware block diagram 70](#_bookmark83)

## [Figure 31 STM32F4 DISCOVERY board 70](#_bookmark84)

## [Figure 32 STM32 ST Link interface 71](#_bookmark86)

## [Figure 33 Driver update for STM board 71](#_bookmark87)

## [Figure 34 Keil C pack installer 72](#_bookmark90)

## [Figure 35 Keil C IDE interface 73](#_bookmark91)

## [Figure 36 Keil C Debug Interface 73](#_bookmark92)

## [Figure 37 ST Cube Mx GUI 74](#_bookmark94)

## [Figure 38 Clock configuration with STM32 Cube Mx 74](#_bookmark95)

## [Figure 39 Project management with ST Cube Mx 74](#_bookmark96)

## [Figure 40 Schematic of Audio peripherals connection on STM32F4 discovery 76](#_bookmark99)

## [Figure 41 Audio playback and record architecture 76](#_bookmark100)

## [Figure 42 Audio playback application flow chart 77](#_bookmark102)

## [Figure 43 Audio recording flow chart 78](#_bookmark104)

## [Figure 44 Setup MCU pin for external trigger 78](#_bookmark106)

## [Figure 45 Volume control diagram 79](#_bookmark107)

**Danh Mục Bảng Biểu**

## [Table 1 FAT file system structure 20](#_bookmark17)

[**Table 2 Advantages and Disadvantages of USB 64**](#_bookmark70)

[**Table 3 Wave File Header 65**](#_bookmark75)

[**Table 4 Wav Data chunk 66**](#_bookmark78)

Embedded system is now can be found in every equipment in our life from simple device as refrigerator, microwave oven … to high technology entertainment devices as smart TV, smart phone or automotive system as engine control system, in-car entertainment system…The heart of these systems is microcontroller. In the role of an Embedded system developer, knowledges about microcontroller architecture and C coding skills are very necessary and important. In order to prepare the necessary knowledge for working as an embedded software developer, I decided to work on the project “Audio playback and recorder using STM32F4 Discovery”.

In this project, I will use development kit STM32F4 Discovery to record sound signal, store the data to an USB stick in the WAV file format and playback recorded signal via a speaker of a headphone. In order to finish this project, I will have to study about many peripherals such as USB to store data then transmit its to sound process chipset to output the sound, I2S to transmit the data to DAC module, DAC in charge of converting digital signal to analog signal, I2C to control the sound process chipset, DMA to allow us to transmit data directly from memory to I2S peripheral in order to reduce the precious clock cycles of CPU. I also have to learn how to associate all above peripherals, ensure every thing can work correctly to create one application.

# Chương I: Tóm Tắt Đồ Án

## Tóm tắt đồ án

Đồ án “Lập trình thiết bị thu âm và phát nhạc với STM32F4” là đề tài ứng dụng kiến thức lập trình và kiến trúc vi điều khiển để tạo ra một đề tài cụ thể.

Trong thời đại cách mạng khoa học và công nghiệp 4.0, cuộc sống con người càng ngày càng yêu cầu nhiều ứng dung như nhà thông minh( cửa tự động, hệ thống tưới nước, tự dộng điều chỉnh, điều khiển giọng nói, camera an ninh…), điện thoại thông minh, hệ thống tự động hóa công nghệ cao trong ô tô như cảnh báo vật cản, tự dộng đỗ xe… Tất cả các ứng dụng trên đều có chung một đặc điểm. Thành phần quan trọng nhất của các hệ thống này chính là các chip vi điều khiển.

Vi điều khiển là một hệ thống rất phức tạp nhưng cũng rất thú vị để tìm hiểu. Với chi phí thấp, năng lượng tiêu thụ thấp cùng những tính năng đa dạng, mạnh mẽ, vi điều khiển xuất hiện trong mọi thiết bị điện tử. Nó cho phép chúng ta thực thi rất nhiều tính năng như thu thập dữ liệu tương tự từ cảm biến, chuyển đổi ra dạng số để xử lý hoặc lưu trữ, điều khiển tốc độ, chiều quay động cơ với bộ điều chế xung, truyền nhận dữ liệu với các giao thức truyền thông như I2C, SPI, UART, tạo trễ và trigger với timer, các chức năng mạng với CAN hoặc Ethenet…

Nhận thức được tầm quan trọng của vi điều khiển trong mỗi hệ thống nhúng, em thực hiện đồ án này nhằm mục tiêu nâng cao kiến thức về kiến trúc vi điều khiển và kỹ năng lập trình C.

## Mục đích nghiên cứu

Như phân tích bên trên, vi điều khiển đóng vai trò rất quan trọng trong hệ thống nhúng nên kiến thức về vi điều khiển là bắt buộc với bất kỳ nhà phát triển phần mềm nhúng nào. Trong những ứng dụng sử dụng rất nhiều ngoại vi, lập trình viên có thể có được những kiến thức rất quý báu về cả vi điều khiển và kỹ năng lập trình.

Có rất nhiều loại vi điều khiển , nhiều kiến trúc. Dựa trên độ dài thanh ghi, ta có vi điều khiển 8 bit, 16 bit, 32 bit. Dựa trên kiến trúc core, chúng ta có vi điều khiển ARM, MIPS, PowerPC.

Trong số các kiên trúc trên , kiến trúc ARM 32 bit là phổ biến nhất hiện nay vì tiết kiệm năng lượng( thích hợp ứng dụng IoT hay thiết bị cầm tay), giá rẻ và khả năng đáp ứng hầu hết các yêu cầu của hệ thống IoT, tự dộng hóa, công nghiệp…

Trong đồ án này, tôi thực thi một số chức năng được liệt kê dưới đây:

* Đọc và ghi dữ liệu định dạng WAV vào thẻ nhớ SD với định dạng FAT32 thông qua USB OTG
* Thu thập dữ liệu tương tự (tín hiệu âm thanh) thông qua mic thu âm.
* Chuyển đổi tín hiệu từ tương tự sang số để chuyển đổi.
* Chuyển đổi tín hiệu từ số ra tương tự để phát nhạc qua loa.
* Sử dụng DMA để nâng cao tốc độ truyền dữ liệu.

## Phương pháp nghiên cứu

Để thực thi các tính năng trên, tôi xác định các yêu cầu của hệ thống như sau:

* + - Đọc tài liệu thiết kế vi điều khiển để hiểu cấu tạo phần cứng của các ngoại vi cần sử dụng.
    - Đọc các tài liệu trên mạng để hiểu các giao thức truyền thông như I2C, I2S…
    - Đọc datasheet của các cảm biến để biết cấu tạo, hoạt động và cách chúng được kết nối đến vi điều khiển.
    - Lập trình, nạp code xuống kit phát triển, debug và kiểm thử.

## Conclusion

Trong chương này, tôi đã trình bày khái quá đề tài đồ án, lý do tại sao tôi thực hiện đồ án. Trong chương tiếp theo, tôi sẽ trình bày về lý thuyết, cấu túc phần cứng và cách hoạt động của các ngoại vi trên vi điều khiển.

# Chương II: Lý Thuyết Tổng Hợp

Trong chương này, tôi sẽ trình bày các nội dung sau:

* FAT32 File system
* Bộ chuyển đổi số tương tự (DAC)
* Phương pháp truy xuất bộ nhớ trực tiếp (DMA)\
* Giao thức truyền thông I2C, SPI/I2S
* Giao thức USB
* Định dạng file WAV

## 2.1 File hệ thống FAT

### **2.1.1 Lịch sử FAT File system**

FAT (File Allocation Table) là kiến trúc hệ thống file và thư mục và họ các quy chuẩn về file để sử dụng nó.

Hệ thống tệp FAT là một hệ thống tệp kế thừa đơn giản và mạnh mẽ. Nó cung cấp hiệu suất tốt với các hệ thống cần lưu trữ tệp nhẹ, đơn giản, nhưng không thể cung cấp hiệu suất, độ tin cậy và khả năng mở rộng tương tự như một số hệ thống tệp hiện đại. Tuy nhiên, nó được hỗ trợ vì lý do tương thích bởi gần như tất cả hệ điều hành hiện đang phát triển cho máy tính cá nhân và nhiều máy tính gia đình, thiết bị di động và hệ thống nhúng, và do đó là định dạng tương thích để trao đổi dữ liệu giữa các máy tính và thiết bị ở hầu hết mọi loại thiết bị phát triển từ năm 1981 đến hiện tại.

Được thiết kế ban đầu vào năm 1977 để sử dụng trên các đĩa mềm, FAT đã sớm được điều chỉnh và sử dụng gần như phổ biến trên các đĩa cứng trong suốt thời đại của hệ điều hành DOS và Windows 9x trong hai thập kỷ. Ngày nay, các hệ thống tệp FAT vẫn thường được tìm thấy trên các đĩa mềm, thẻ nhớ USB, flash và các thẻ nhớ và mô-đun trạng thái rắn khác, cùng nhiều thiết bị di động và nhúng.

### 2.1.2 Tổng quan về hệ thống tệp FAT

Tên của hệ thống tệp bắt nguồn từ việc sử dụng nổi bật của hệ thống tệp của bảng chỉ mục, Bảng phân bổ tệp, được phân bổ tĩnh tại thời điểm định dạng. Bảng chứa các mục nhập cho mỗi cụm, một vùng lưu trữ đĩa liền kề. Mỗi mục chứa số cụm tiếp theo trong tệp hoặc một dấu hiệu khác chỉ ra phần cuối của tệp, không gian đĩa không sử dụng hoặc các khu vực dành riêng đặc biệt của đĩa. Thư mục gốc của đĩa chứa số cụm đầu tiên của mỗi tệp trong thư mục đó; hệ điều hành sau đó có thể duyệt qua bảng FAT, tìm kiếm số cụm của từng phần liên tiếp của tệp đĩa dưới dạng chuỗi cụm cho đến khi kết thúc tệp. Theo cách tương tự, các thư mục con được triển khai dưới dạng các tệp.

Được thiết kế ban đầu như một hệ thống tệp 8 bit, số lượng cụm tối đa đã được tăng lên đáng kể khi các ổ đĩa đã phát triển, và do đó số lượng bit được sử dụng để xác định từng cụm đã tăng lên. Các phiên bản chính liên tiếp của định dạng FAT được đặt tên theo số bit phần tử bảng: 12 (FAT12), 16 (FAT16) và 32 (FAT32). Ngoại trừ tiền thân FAT 8 bit ban đầu, mỗi biến thể này vẫn đang được sử dụng. Tiêu chuẩn FAT cũng đã được mở rộng theo những cách khác nhưng nói chung vẫn duy trì khả năng tương thích với phần mềm hiện có.

### **2.1.3 Cấu trúc phân vùng hệ thống tệp FAT**

Tổng quan về tổ chức phân vùng trong hệ thống tệp FAT

*Bảng 1 Cấu trúc hệ thống tệp FAT*

|  |  |  |
| --- | --- | --- |
| **Phân vùng** | **Kích thước(sector)** | **Nội dung** |
| Sectors dự phòng | number of reserved sectors | Boot Sector |
| Sector về thông tin hệ thống tệp (chỉ FAT32) |
| sectors dự phòng khác(tùy chọn) |
| Vùng  FAT | (Số bảng FATs) \* (số sectors trên một bảng FAT) | Bảng phân bố file #1 |
| Bảng phân bố file #2 ... (tùy chọn) |
| Phân vùng thư mục gốc | (Số thư mục gốc  \* 32) / (số bytes trên sector) | Thư mục gốc (chỉ FAT12 và FAT16) |
| Phân vùng dữ liệu | (số cụm)\* (số sectors trên cụm) | Phân vùng dữ liệu (cho file hay thư mục) ... (cho đến khi kết thúc đĩa cứng) |

**Sectors dự phòng**

Phân vùng này chứa một số thông tin hệ thống tệp cơ bản, đặc biệt là loại và con trỏ tới vị trí của các phần khác và thường chứa mã bộ tải khởi động của hệ điều hành.

Riêng với FAT32, phân vùng này còn chưa thêm sector về thông tin hệ thống file ở vị trí sector 1 và một boot sector dự phòng ở vị trí sector thứ 6.

**Phân vùng FAT**

Đây là phân vùng chứa 2 bảng FAT, một bản chính, một bản dự phòng dù rất ít khi dùng, ngay cả với các tiện ích sửa chữa đĩa. Hai bảng FAT này là bản đồ của cả ổ cứng hay vùng nhớ. Nó chỉ ra cụm nào được sử dụng bởi file hay thư mục nào, từ đó hệ điều hành có thể tìm tiếm và

thực hiện các tác vụ với file hay thư mục.

**Phân vùng thư mục gốc**  
Đây là Bảng thư mục lưu trữ thông tin về các tệp và thư mục nằm trong thư mục gốc. Nó chỉ được sử dụng với FAT12 và FAT16 và áp đặt cho thư mục gốc một kích thước tối đa cố định được phân bổ trước khi tạo tập này. FAT32 lưu trữ thư mục gốc trong Vùng dữ liệu, cùng với các tệp và các thư mục khác, cho phép nó phát triển mà không bị ràng buộc như vậy. Do đó, đối với FAT32, Vùng dữ liệu bắt đầu tại đây

**Vùng dữ liệu**

Đây là nơi lưu trữ dữ liệu tệp và thư mục thực tế và chiếm phần lớn phân vùng.

Kích thước của tệp và thư mục con có thể được tăng tùy ý (miễn là có các cụm miễn phí) bằng cách thêm nhiều liên kết vào chuỗi của tệp trong FAT. Tuy nhiên, lưu ý rằng các tệp được phân bổ theo đơn vị cụm, vì vậy nếu tệp 1 KB nằm trong cụm 32 KB, 31 KB bị lãng phí.

FAT32 thường có bảng thư mục gốc bắt đầu trong cụm thứ 2: cụm đầu tiên của phân vùng dữ liệu.

Fat sử dụng định dạng little edian cho các entry trong phần header và bảng FAT. Nó có thể cấp phát nhiều FAT sector hơn cho một số lượng cluster.

## 2.2Bộ chuyển đổi số tương tự DAC

Trong điện tử, bộ chuyển đổi tín hiệu số sang tương tự (DAC, D / A, D2A hoặc D-to-A) là một hệ thống chuyển đổi tín hiệu số thành tín hiệu tương tự. Bộ chuyển đổi tương tự sang số (ADC) thực hiện chức năng đảo ngược.

Có một số kiến ​​trúc DAC; sự phù hợp của một bộ xử lý tín hiệu cho một ứng dụng cụ thể được xác định bằng các tham số bao gồm: độ phân giải, tần suất lấy mẫu tối đa và các thứ khác. Chuyển đổi từ kỹ thuật số sang tương tự có thể làm giảm tín hiệu, do đó, một bộ định tuyến phải được chỉ định có lỗi không đáng kể về mặt ứng dụng.

Các bộ xử lý tín hiệu thường được sử dụng trong các máy nghe nhạc để chuyển đổi luồng dữ liệu số thành tín hiệu âm thanh analog. Chúng cũng được sử dụng trong TV và điện thoại di động để chuyển đổi dữ liệu video kỹ thuật số thành tín hiệu video tương tự kết nối với trình điều khiển màn hình để hiển thị màu đơn sắc hoặc ảnh màu.

Hai ứng dụng này sử dụng các bộ xử lý tín hiệu ở hai đầu đối diện của sự hy sinh tần số hoặc độ phân giải. Bộ xử lý âm thanh là loại tần số thấp, độ phân giải cao trong khi bộ xử lý video là loại có độ phân giải thấp đến trung bình.

Do sự phức tạp và nhu cầu về các thành phần được kết hợp chính xác, tất cả trừ các bộ vi xử lý chuyên dụng nhất được triển khai dưới dạng các mạch tích hợp (IC). Các bộ xử lý rời rạc thường có tốc độ cực cao, độ phân giải thấp, các loại ngốn điện, như được sử dụng trong các hệ thống radar quân sự. Thiết bị thử nghiệm tốc độ rất cao, đặc biệt là máy hiện sóng lấy mẫu, cũng có thể sử dụng các bộ vi xử lý rời rạc.

Một bộ chuyển đổi chuyển đổi một số chính xác thành một đại lượng vật lý (ví dụ: điện áp hoặc áp suất). Cụ thể, các bộ xử lý tín hiệu thường được sử dụng để chuyển đổi dữ liệu chuỗi thời gian có độ chính xác hữu hạn thành tín hiệu vật lý liên tục thay đổi.

Một bộ xử lý lý tưởng chuyển đổi các số trừu tượng thành một chuỗi các xung sau đó được xử lý bởi bộ lọc và được tái tạo bằng cách sử dụng một số dạng nội suy để điền dữ liệu giữa các xung. Một bộ xử lý thực tế thông thường chuyển đổi các số thành một hàm hằng số được tạo thành từ một chuỗi các hàm hình chữ nhật được mô hình hóa với việc giữ không thứ tự. Các phương pháp DAC khác (chẳng hạn như các phương pháp dựa trên điều chế delta-sigma) tạo ra đầu ra điều chế mật độ xung có thể được lọc tương tự để tạo ra tín hiệu thay đổi trơn tru.

Theo lý thuyết lấy mẫu Nyquist-Shannon, một bộ DAC có thể tái tạo tín hiệu gốc từ dữ liệu được lấy mẫu với điều kiện băng thông của nó đáp ứng một số yêu cầu nhất định (ví dụ: tín hiệu băng cơ sở có băng thông nhỏ hơn một nửa tần số Nyquist).

### **2.2.1 Bộ chuyển đổi số tương tự trên vi điều khiển STM32F407VG**

Mô-đun DAC là bộ chuyển đổi kỹ thuật số sang tương tự 12 bit, điện áp đầu ra. Bộ xử lý có thể được cấu hình ở chế độ 8- hoặc 12 bit và có thể được sử dụng cùng với bộ điều khiển DMA. Ở chế độ 12 bit, dữ liệu có thể được căn trái hoặc phải. Bộ xử lý tín hiệu có hai kênh đầu ra, mỗi kênh có bộ chuyển đổi riêng. Trong chế độ kênh DAC kép, chuyển đổi có thể được thực hiện độc lập hoặc đồng thời khi cả hai kênh được nhóm lại với nhau cho các hoạt động cập nhật đồng bộ. Một pin tham chiếu đầu vào, VREF + (được chia sẻ với ADC) có sẵn để làm cơ sở chuyển đổi mức tín hiệu.

Các tính năng chính của DAC:

• Hai bộ chuyển đổi DAC: mỗi kênh một đầu ra

• Căn chỉnh dữ liệu trái hoặc phải trong chế độ 12 bit

• Kh năng cập nhật được đồng bộ hóa.

• Tạo sóng nhiễu

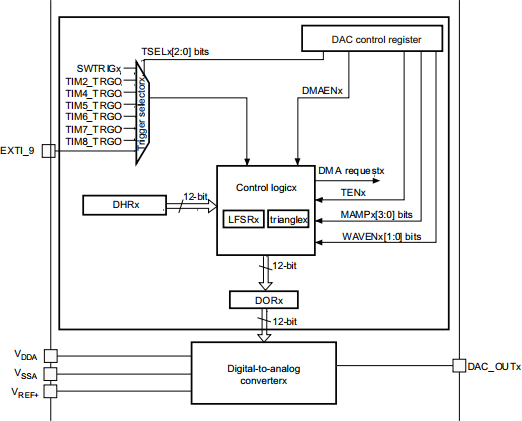
• Tạo sóng tam giác

• Kênh DAC kép để chuyển đổi độc lập hoặc đồng thời

• Khả năng truy cập bộ nhớ trực tiếp cho từng kênh

• Kích hoạt bên ngoài để chuyển đổi

• Tham chiếu điện áp đầu vào, VREF +



*Ảnh 1 Sơ đồ khối DAC*

### **2.2.2 Các bước khởi tạo và sử dụng DAC**

Để chuyển đổi dữ liệu số sang dữ liệu tương tự với mô-đun DAC:

• Khởi tạo đầu vào, đầu ra, đất, điện áp tham chiếu cho DAC

• Chọn nguồn kích hoạt cho mô-đun: timer nội, pin bên ngoài hoặc trigger phần mềm

• Định cấu hình các tham số (yêu cầu DMA, tạo xung tam giác, bật ngắt DMA, biên độ dao động) để xác định chế độ hoạt động của mô-đun DAC.

• Kích hoạt ISR kênh DMA để quản lý hoạt động của mô-đun DAC qua ngắt

## 2.3 Cơ chế truy cập bộ nhớ trực tiếp (DMA)

Truy cập bộ nhớ trực tiếp (DMA) là một tính năng của các hệ thống máy tính cho phép một số hệ thống con phần cứng nhất định truy cập bộ nhớ hệ thống chính (bộ nhớ truy cập ngẫu nhiên), độc lập với bộ xử lý trung tâm (CPU).

Không có DMA, khi CPU đang sử dụng đầu vào / đầu ra được lập trình, nó thường bị chiếm dụng hoàn toàn trong toàn bộ thời gian của hoạt động đọc hoặc ghi và do đó không thẻ để thực hiện công việc khác. Với DMA, CPU trước tiên bắt đầu truyền, sau đó nó thực hiện các hoạt động khác trong khi quá trình truyền đang diễn ra và cuối cùng nó nhận được một ngắt từ bộ điều khiển DMA (DMAC) khi hoạt động được thực hiện. Tính năng này hữu ích bất cứ lúc nào CPU không theo kịp tốc độ truyền dữ liệu hoặc khi CPU cần thực hiện công việc trong khi chờ truyền dữ liệu I / O tương đối chậm. Nhiều hệ thống phần cứng sử dụng DMA, bao gồm bộ điều khiển ổ đĩa, card đồ họa, card mạng và card âm thanh. DMA cũng được sử dụng để truyền dữ liệu trong chip xử lý đa lõi. Máy tính có kênh DMA có thể truyền dữ liệu đến và từ các thiết bị mà ít sử dụng CPU hơn so với máy tính không có kênh DMA. Tương tự, một phần tử xử lý bên trong bộ xử lý đa lõi có thể truyền dữ liệu đến và từ bộ nhớ cục bộ của nó mà không cần dùng tới những xung đồng hồ quý giá của CPU. Điều này cho phép việc tính toán và đọc ghi dữ liệu được thức hiện song song.

DMA cũng có thể được sử dụng để sao chép "bộ nhớ vào bộ nhớ" hoặc di chuyển dữ liệu trong bộ nhớ.

### **2.3.1 Nguyên lý truy cập bộ nhớ trực tiếp**

DMA tiêu chuẩn, còn được gọi là DMA của bên thứ ba, sử dụng bộ điều khiển DMA. Bộ điều khiển DMA có thể tạo địa chỉ bộ nhớ và khởi tạo chu kỳ đọc hoặc ghi bộ nhớ. Nó chứa một số thanh ghi phần cứng có thể được ghi và đọc bởi CPU. Chúng bao gồm một thanh ghi địa chỉ bộ nhớ, một thanh ghi đếm byte và một hoặc nhiều thanh ghi điều khiển. Tùy thuộc vào tính năng nào mà bộ điều khiển DMA cung cấp, các thanh ghi điều khiển này có thể chỉ định một số kết hợp nguồn, đích, hướng chuyển (đọc từ thiết bị I / O hoặc ghi vào thiết bị I / O), kích thước của đơn vị truyền và số byte cần truyền trong một cụm.

Để thực hiện thao tác nhập, xuất hoặc bộ nhớ vào bộ nhớ, bộ xử lý CPU khởi tạo bộ điều khiển DMA với số lượng từ cần truyền và địa chỉ bộ nhớ sẽ sử dụng. CPU sau đó ra lệnh cho thiết bị ngoại vi để bắt đầu truyền dữ liệu. Bộ điều khiển DMA sau đó cung cấp địa chỉ và các dòng điều khiển đọc / ghi vào bộ nhớ hệ thống. Mỗi khi một byte dữ liệu sẵn sàng được chuyển giữa thiết bị ngoại vi và bộ nhớ, bộ điều khiển DMA sẽ tăng thanh ghi địa chỉ bên trong cho đến khi toàn bộ khối dữ liệu được truyền hết.

### **2.3.2 Vấn đề quản lý đường Bus**

Trong hệ thống DMA của bên thứ nhất, mỗi CPU và các thiết bị ngoại vi có thể được cấp quyền điều khiển bus bộ nhớ. Khi một thiết bị ngoại vi có thể trở thành bus chính, nó có thể ghi trực tiếp vào bộ nhớ hệ thống mà không cần sự tham gia của CPU, cung cấp địa chỉ bộ nhớ và tín hiệu điều khiển theo yêu cầu. Một số biện pháp phải được cung cấp để đưa bộ xử lý tranh chấp quyền sử dụng bus với ngoại vi.

### **2.3.3 Các chế độ truyền dữ liệu**

Việc truyền dữ liệu qua DMA có thể chuyển từng byte một lần hoặc tất cả cùng một lúc trong chế độ truyền liên tục. Nếu truyền một byte tại một thời điểm, CPU có thể truy cập bộ nhớ vào các chu kỳ bus xen kẽ.

Trong chế độ truyền liên tục, DMA giữ quyền kiểm soát đường bus và điều khiển bộ nhớ để truyền đi hang tram hay hang triệu byte một cách liên tCPU sẽ không thể sử dụng các tài nguyên này trong thời gian đó.

**2.3.4 Các chế độ hoạt động của DMA**

**Chế độ truyền liên tục:**

Trong chế độ chụp liên tục, toàn bộ khối dữ liệu được truyền theo một chuỗi liền kề. Khi bộ điều khiển DMA được CPU cấp quyền truy cập vào bus hệ thống, nó sẽ chuyển tất cả byte dữ liệu trong khối dữ liệu trước khi giải phóng điều khiển của bus hệ thống trở lại CPU, nhưng khiến CPU không hoạt động trong thời gian tương đối dài. Chế độ này còn được gọi là "Chế độ chuyển khối".

**Chế độ truyền từng byte:**

Chế độ truyền từng byte được sử dụng trong các hệ thống không nên tắt CPU trong khoảng thời gian cần thiết cho các chế độ truyền phát. Trong chế độ truyền từng byte, bộ điều khiển DMA có được quyền truy cập vào bus hệ thống giống như ở chế độ truyền liên tục, sử dụng tín hiệu BR (Bus Request) và BG (Bus Grant), là hai tín hiệu điều khiển giao diện giữa CPU và Bộ điều khiển DMA. Tuy nhiên, trong chế độ truyền từng byte, sau một byte truyền dữ liệu, việc điều khiển bus hệ thống được xác nhận lại cho CPU thông qua BG. Sau đó, nó liên tục được yêu cầu lại thông qua BR, truyền một byte dữ liệu cho mỗi yêu cầu, cho đến khi toàn bộ khối dữ liệu được truyền. Bằng cách liên tục có được và giải phóng điều khiển bus hệ thống, bộ điều khiển DMA về cơ bản xen kẽ các lệnh truyền và truyền dữ liệu. CPU xử lý một lệnh, sau đó bộ điều khiển DMA truyền một giá trị dữ liệu, v.v. Một mặt, khối dữ liệu không được truyền nhanh trong chế độ truyền từng byte theo chu kỳ như ở chế độ truyền liên tục, nhưng mặt khác, CPU không được nhàn rỗi lâu như ở chế độ truyền liên tục. Chế độ truyền từng byte rất hữu ích cho các bộ điều khiển theo dõi dữ liệu trong thời gian thực.

**Chế độ hỗn hợp:**

Chế độ hỗn hợp mất nhiều thời gian nhất để truyền một khối dữ liệu, tuy nhiên nó cũng là chế độ hiệu quả nhất về mặt hiệu suất hệ thống tổng thể. Ở chế độ hỗn hợp, bộ điều khiển DMA chỉ truyền dữ liệu khi CPU đang thực hiện các hoạt động không sử dụng các bus hệ thống. Ưu điểm chính của chế độ hỗn hợp là CPU không bao giờ ngừng thực thi các chương trình của nó và việc chuyển DMA là miễn phí về thời gian, trong khi nhược điểm là phần cứng cần xác định khi CPU không sử dụng các bus hệ thốn. Điều này đôi khi rất phức tạp.

### **2.3.5 DMA mo-dun tích hợp trên STM32F407VG**

Các tính năng chính :

• Kiến trúc bus chính AHB kép, một dành riêng cho truy cập bộ nhớ và một dành riêng cho truy cập ngoại vi

• Giao diện lập trình AHB chỉ hỗ trợ truy cập 32 bit

• 8 luồng cho mỗi bộ điều khiển DMA, tối đa 8 kênh (yêu cầu) trên mỗi luồng

* + Mỗi luồn có thể được cài đặt để:

- một kênh thông thường hỗ trợ chuyển từ bộ nhớ ngoại vi sang bộ nhớ, bộ nhớ đến ngoại vi và bộ nhớ sang bộ nhớ

- kênh song song cũng hỗ trợ bộ đệm song song ở phía bộ nhớ

• Mỗi trong số 8 luồng được kết nối với các kênh (yêu cầu) phần cứng chuyên dụng

• Ưu tiên giữa các yêu cầu luồng DMA là lập trình phần mềm (4 cấp độ bao gồm rất cao, cao, trung bình, thấp) hoặc phần cứng trong trường hợp bình đẳng (yêu cầu 0 được ưu tiên hơn yêu cầu 1, v.v.),

• Mỗi luồng cũng hỗ trợ kích hoạt phần mềm để chuyển từ bộ nhớ sang bộ nhớ (chỉ khả dụng cho bộ điều khiển DMA2)

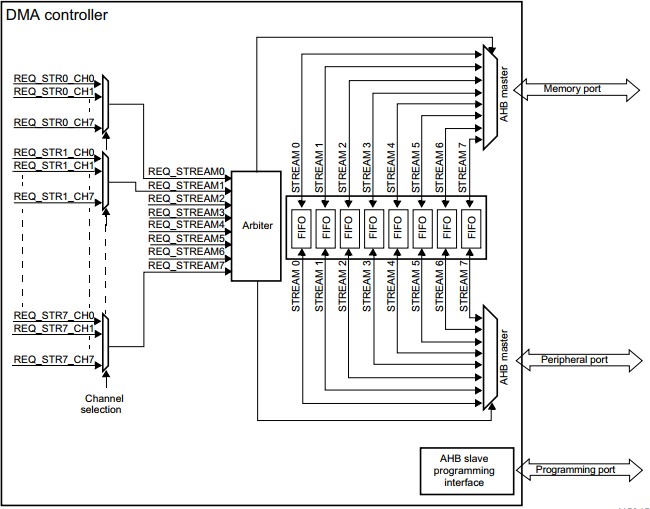
• Mỗi yêu cầu luồng có thể được chọn trong số tối đa 8 yêu cầu kênh có thể. Lựa chọn này có thể cấu hình bằng phần mềm và cho phép một số thiết bị ngoại vi khởi tạo DMA yêu cầu

• Độ rộng truyền nguồn và đích độc lập (byte, nửa từ, từ): khi độ rộng dữ liệu của nguồn và đích không bằng nhau, DMA sẽ tự động đóng gói / giải nén các dữ liệu cần thiết để tối ưu hóa băng thông. Tính năng này chỉ khả dụng trong chế độ FIFO

• Tăng địa chỉ hoặc không tăng địa chỉ cho nguồn và đích

• 5 cờ sự kiện (Chuyển một nửa DMA, Hoàn thành chuyển DMA, Lỗi chuyển DMA, Lỗi DMA FIFO, Lỗi chế độ trực tiếp) hợp lý với nhau trong một yêu cầu ngắt cho mỗi luồng

### **2.3.6 Chi tiết chức năng của DMA**



*Ảnh 6 Sơ đồ khối DMA*

Bộ điều khiển DMA thực hiện chuyển bộ nhớ trực tiếp: khi có quyền khiểm soát bus AHB, nó có thể điều khiển ma trận bus AHB để bắt đầu các hoạt động truyền dữ liệu sử dụng bus AHB.

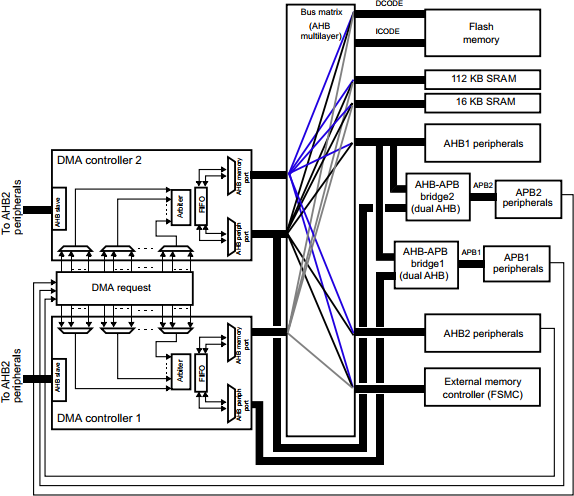
Nó có thể thực hiện các hướng chuyển dữ liệu sau:

• bộ nhớ ngoại vi

• bộ nhớ ngoại vi

• bộ nhớ đến bộ nhớ

Bộ điều khiển DMA cung cấp hai cổng chính AHB: cổng bộ nhớ AHB, dự định được kết nối với bộ nhớ và cổng ngoại vi AHB, dự định được kết nối với các thiết bị ngoại vi. Tuy nhiên, để cho phép chuyển bộ nhớ sang bộ nhớ, cổng ngoại vi AHB cũng phải có quyền truy cập vào bộ nhớ.



*Ảnh 7 Cấu tạo phần cứng trên vi điều khiển STM32F407*

### **2.3.7 Quá trình truyền dữ liệu với DMA**

Việc truyền dữ liệu qua DMA bao gồm một chuỗi các lần chuyển dữ liệu nhất định. Số lượng các mục dữ liệu được truyền và chiều rộng của chúng (8 bit, 16 bit hoặc 32 bit) được lập trình phần mềm. Mỗi lần chuyển DMA bao gồm ba thao tác:

• Tải từ thanh ghi dữ liệu ngoại vi hoặc vị trí trong bộ nhớ

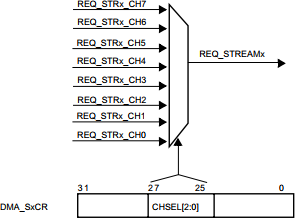
• Lưu trữ dữ liệu được tải vào thanh ghi dữ liệu ngoại vi.

• Giảm số lượng phần tử cần truyền.

Sau một sự kiện, thiết bị ngoại vi sẽ gửi tín hiệu yêu cầu đến bộ điều khiển DMA. Bộ điều khiển DMA phục vụ yêu cầu tùy thuộc vào các ưu tiên của kênh. Ngay khi bộ điều khiển DMA truy cập vào thiết bị ngoại vi, tín hiệu ACK được gửi đến thiết bị ngoại vi bởi bộ điều khiển DMA. Thiết bị ngoại vi giải phóng yêu cầu của nó ngay khi nhận được tín hiệu ACK từ bộ điều khiển DMA. Khi yêu cầu đã được xác nhận lại bởi thiết bị ngoại vi, bộ điều khiển DMA sẽ phát ra tín hiệu ACK.

Lựa chọn kênh truyền:

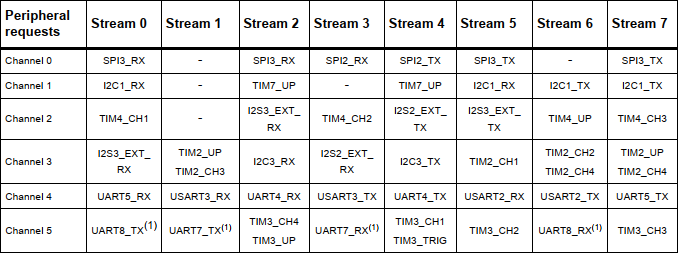
Mỗi luồng được liên kết với một yêu cầu DMA có thể được chọn trong số 8 yêu cầu kênh có thể.



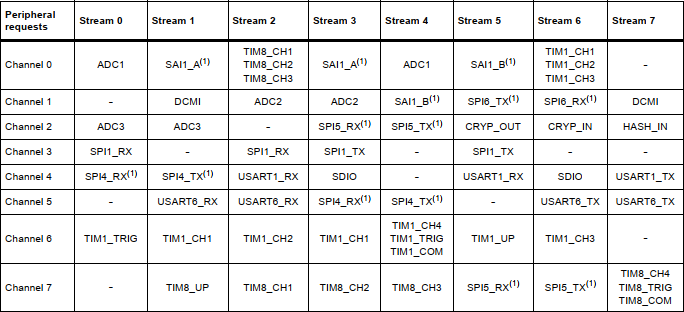
*Ảnh 8 Cơ chế lựa chọn kênh truyền DMA*

8 yêu cầu từ các thiết bị ngoại vi (TIM, ADC, SPI, I2C, v.v.) được kết nối độc lập với từng kênh.

Hai hình dưới đây cho thấy kết nối giữa các kênh DMA với các thiết bị ngoại vi



*Ảnh 9 Kết nối kênh và ngoại vi trên DMA1*



*Ảnh 10 Kết nối kênh và ngoại vi trên DMA2*

**Bộ phân xử**

Có rất nhiều các kênh DMA nối với các ngoại vi mà tại một thời điểm nhất định, chỉ một kênh được hoạt động. Vì vậy cần thiết có một bộ phân xử để quyết định xem kênh nào được phép hoạt động. Việc phân xử này dựa trên mức độ ưu tiên của kênh DMA.

Độ ưu tiên được quản lý dưới hai dạng:

* Phần mềm: mỗi luồng ưu tiên có thể được cấu hình trong thanh ghi điều khiển DMA. Có bốn cấp độ:

- Ưu tiên rất cao

- Ưu tiên cao

- Ưu tiên trung bình

- Ưu tiên thấp

* Phần cứng: Nếu hai yêu cầu có cùng mức ưu tiên phần mềm, luồng có số thấp hơn sẽ ưu tiên hơn luồng có số cao hơn. Ví dụ: Luồng 2 được ưu tiên hơn Luồng 4.

**Các luồng DMA**

Mỗi trong số 8 luồng bộ điều khiển DMA cung cấp một liên kết truyền một chiều giữa nguồn và đích.

Mỗi luồng có thể được cấu hình để thực hiện:

• Truyền dữ liệu thông thường: chuyển từ bộ nhớ sang thiết bị ngoại vi, chuyển từ bộ nhớ sang bộ nhớ hoặc bộ nhớ sang bộ nhớ

• Truyền dữ liệu song song trên hai kênh DMA: chuyển bộ đệm đôi bằng cách sử dụng hai con trỏ bộ nhớ cho bộ nhớ (trong khi DMA đang đọc / ghi từ / đến bộ đệm, ứng dụng có thể ghi / đọc sang / từ bộ đệm khác).

Lượng dữ liệu được truyền (tối đa 65535) có thể lập trình được và liên quan đến độ rộng nguồn của thiết bị ngoại vi yêu cầu chuyển DMA được kết nối với cổng AHB ngoại vi. Thanh ghi chứa số lượng các mục dữ liệu sẽ được chuyển giảm dần sau mỗi lần dữ liệu được truyền đi.

**Nguồn, đích và chế độ truyền dữ liệu**

Việc chuyển dữ liệu giữa nguồn và đích đều có thể xử lý các thiết bị ngoại vi và bộ nhớ trong toàn bộ khu vực 4 GB, tại các địa chỉ nằm trong khoảng từ 0x0000 0000 đến 0xFFFF FFFF.

Hướng được định cấu hình bằng cách sử dụng thanh ghi cụ thể và cung cấp ba khả năng: chuyển từ bộ nhớ sang ngoại vi, ngoại vi sang bộ nhớ hoặc bộ nhớ sang bộ nhớ.

**Địa chỉ nguồn và đích**

Khi chiều rộng dữ liệu lần lượt là một nửa từ hoặc một từ, địa chỉ ngoại vi hoặc bộ nhớ được ghi vào các thanh ghi phải được căn chỉnh trên một ranh giới địa chỉ từ hoặc nửa từ tương ứng.

Trong ứng dụng này, tôi chỉ sử dụng chế độ ngoại vi cho bộ nhớ nên tôi sẽ giải thích chi tiết chế độ DMA này.

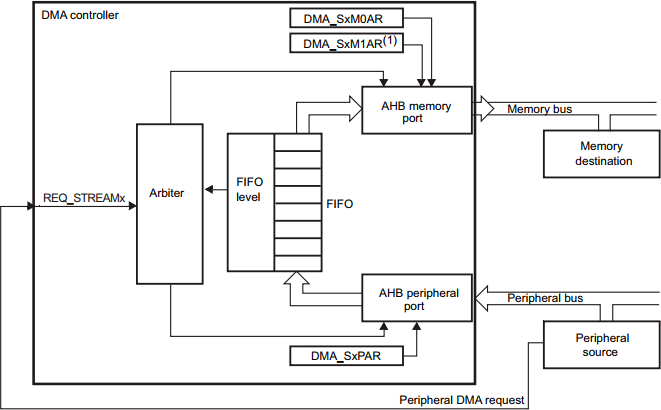
**Chế độ truyền ngoại vi vào bộ nhớ**

Khi chế độ này được bật, mỗi khi có yêu cầu ngoại vi xảy ra, luồng DMA sẽ bắt đầu chuyển dữ liệu từ nguồn để điền vào FIFO.

Khi đạt đến ngưỡng ngưỡng của FIFO, nội dung của FIFO sẽ được truyền đi và được lưu trữ vào đích.

Quá trình truyền dừng lại khi thanh ghi bộ đếm DMA về 0, khi thiết bị ngoại vi yêu cầu kết thúc chuyển (trong trường hợp bộ điều khiển luồng ngoại vi) hoặc khi mô-dun DMA được vô hiệu hóa bằng phần mềm. Trong chế độ trực tiếp, mức ngưỡng của FIFO không được sử dụng: sau mỗi lần truyền dữ liệu duy nhất từ ​​thiết bị ngoại vi sang FIFO, dữ liệu tương ứng sẽ được truyền đi ngay lập tức và được lưu trữ vào đích.

Tại một thời điểm, nếu có nhiều luồng DMA muốn truyền thì luồng có độ ưu tiên cao nhất sẽ được thực hiện việc dịch chuyển dữ liệu



*Ảnh 11 Sơ đồ truyền dữ liệu từ ngoại vi vào bộ nhớ*

**Con trỏ ngoại vi và bộ nhớ**

Con trỏ ngoại vi và bộ nhớ có thể tùy ý tự động tăng sau hoặc giữ nguyên sau mỗi lần chuyển tùy thuộc thanh ghi điều khiển DMA.

Vô hiệu hóa chế độ tăng địa chỉ rất hữu ích khi dữ liệu nguồn hoặc dữ liệu đích được truy cập thông qua một thanh ghi.

Nếu chế độ tự động tăng địa chỉ được bật, địa chỉ của lần chuyển tiếp theo sẽ là địa chỉ của lần chuyển trước tăng thêm 1 (đối với byte), 2 (đối với nửa từ) hoặc 4 (đối với từ) tùy thuộc vào độ rộng dữ liệu được lập trình trong thanh ghi điều khiển DMA.

Để tối ưu hóa hoạt động đóng gói, có thể sửa kích thước bù tăng cho địa chỉ ngoại vi bất kể

kích thước của dữ liệu được truyền trên cổng ngoại vi AHB.

**Chế độ vòng**

Chế độ vòng tronfthichs hợp để xử lý bộ đệm tròn và luồng dữ liệu liên tục (ví dụ: chế độ quét ADC). Khi chế độ vòng tròn được kích hoạt, số lượng mục dữ liệu sẽ được tự động tải lại với giá trị ban đầu được lập trình trong giai đoạn cấu hình luồng và các yêu cầu DMA tiếp tục được phục vụ.

**Chế độ DMA song song**

Chế độ này có sẵn cho tất cả các luồng DMA1 và DMA2.

Chế độ DMA song song hoạt động như một luồng DMA thông thường (bộ đệm đơn) với sự khác biệt là nó có hai con trỏ bộ nhớ. Khi chế độ DMA song song, chế độ truyền vòng tròn sẽ tự động được bật và ở mỗi đầu lần truyền, các con trỏ bộ nhớ được hoán đổi.

Trong chế độ này, bộ điều khiển DMA hoán đổi từ con trỏ bộ nhớ này sang con trỏ bộ nhớ khác ở mỗi đầu giao dịch. Điều này cho phép phần mềm xử lý một vùng nhớ trong khi vùng nhớ thứ hai được lấp đầy / sử dụng bởi chuyển DMA. Luồng DMA song song có thể hoạt động ở cả hai hướng (bộ nhớ có thể là nguồn hoặc đích).

## 2.4Giao thức SPI/I2S

### **2.4.1 Giới thiệu giao thức SPI/I2S**

SPI (tiếng Anh: Serial Peripheral Interface, SPI bus — Giao diện Ngoại vi Nối tiếp, bus SPI) là một chuẩn đồng bộ nối tiếp để truyền dữ liệu ở chế độ song công toàn phần full-duplex, do công ty Motorola thiết kế nhằm đảm bảo sự liên hợp giữa các vi điều khiển và thiết bị ngoại vi một cách đơn giản và giá rẻ. Đôi khi SPI còn được gọi là giao diện bốn-dây

Giao diện SPI cung cấp hai chức năng chính, hỗ trợ giao thức SPI hoặc giao thức âm thanh I2S. Theo mặc định, đó là chức năng SPI được chọn. Có thể chuyển giao diện từ SPI sang I2S bằng phần mềm.

Khác với cổng nối tiếp chuẩn (tiếng Anh: standard serial port), SPI là giao diện đồng bộ, trong đó bất cứ quá trình truyền nào cũng được đồng bộ hóa với tín hiệu đồng hồ chung, tín hiệu này sinh ra bởi thiết bị chủ động (bộ vi xử lý). Thiết bị ngoại vi bên phía nhận (bị động) làm đồng bộ quá trình nhận chuỗi bit với tín hiệu đồng hồ. Có thể kết nối một số vi mạch vào mỗi giao diện ngoại vi nối tiếp của vi mạch-thiết bị chủ động. Thiết bị chủ động chọn thiết bị động để truyền dữ liệu bằng cách kích hoạt tín hiệu "chọn chip" (tiếng Anh: chip select) trên vi mạch bị động. Thiết bị ngoại vi nếu không được chọn bởi bộ vi xử lý sẽ không tham gia vào quá trình truyền theo giao diện SPI.

Trong giao diện SPI có sử dụng bốn tín hiệu số:

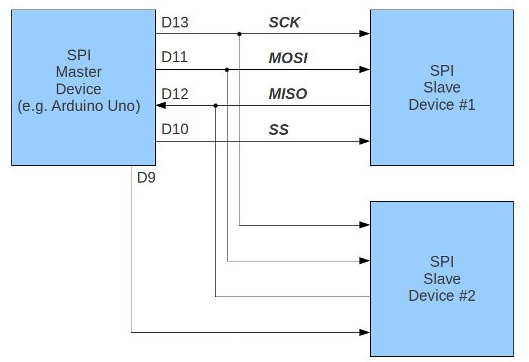
MOSI hay SI — cổng ra của bên chủ động, cổng vào của bên bị động (tiếng Anh: Master Out Slave In), dành cho việc truyền dữ liệu từ thiết bị chủ động đến thiết bị động.

MISO hay SO — cổng vào của bên chủ động, cổng ra của bên bị động (tiếng Anh: Master In Slave Out), dành cho việc truyền dữ liệu từ thiết bị động đến thiết bị chủ động.

SCLK hay SCK — tín hiệu đồng hồ nối tiếp (tiếng Anh: Serial Clock), dành cho việc truyền tín hiệu đồng hồ cho thiết bị động.

CS hay SS — chọn vi mạch, chọn bên bị động (tiếng Anh: Chip Select, Slave Select).

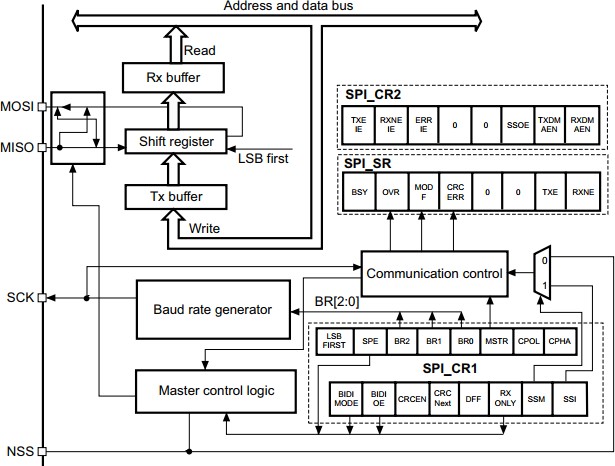
I2S cũng là một giao diện truyền thông nối tiếp đồng bộ. Nó có thể giải quyết bốn tiêu chuẩn âm thanh khác nhau bao gồm tiêu chuẩn I2S Philips, tiêu chuẩn MSB và LSB và tiêu chuẩn PCM. Nó có thể hoạt động như một thiết bị bị động hoặc một thiết bị chính ở chế độ song công hoàn toàn (sử dụng 4 chân) hoặc ở chế độ bán song công (sử dụng 3 chân).



Sơ đồ kết nối SPI/I2S

### **2.4.2 Chi tiết chức năng SPI**

**Giới thiệu chung**



*Ảnh 15 Sơ đồ khối SPI*

Thông thường, SPI được kết nối với các thiết bị bên ngoài thông qua bốn chân:

• MISO: Master In / Slave Out dữ liệu. Pin này có thể được sử dụng để truyền dữ liệu ở chế độ slave và nhận dữ liệu ở chế độ master.

• MOSI: Master Out / Slave In data. Pim này có thể được sử dụng để truyền dữ liệu ở chế độ master và nhận dữ liệu ở chế độ slave.

• SCK: Đầu ra xung đồng hồ nối tiếp cho các master và đầu vào cho các thiết bị slave.

• NSS: Chọn thiết bị slave. Đây là một pin tùy chọn để chọn một thiết bị slave. Chốt này hoạt động như một con chip chọn để cho phép thiết bị master liên lạc với các thiết bị slave riêng lẻ và để tránh sự tranh chấp trên các dòng dữ liệu. Đầu vào NSS slave có thể được điều khiển bởi các cổng IO tiêu chuẩn trên thiết bị master. Chân NSS cũng có thể được sử dụng làm đầu ra nếu được bật và được điều khiển ở mức thấp nếu SPI ở chế độ master. Theo cách này, tất cả các chân NSS từ các thiết bị được kết nối với chân Master NSS đều thấy mức thấp và trở thành slave khi chúng được cấu hình ở chế độ NSS phần cứng.

Các chân MOSI được kết nối với nhau và các chân MISO được kết nối với nhau. Theo cách này, dữ liệu được truyền tuần tự giữa master và slave (theo thứ tự MSB).

Việc giao tiếp luôn được bắt đầu bởi master. Khi thiết bị master truyền dữ liệu đến thiết bị slave thuộc thông qua chân MOSI, thiết bị slave sẽ phản hồi thông qua chân MISO. Điều này ngụ ý giao tiếp song công hoàn toàn với cả dữ liệu ra và dữ liệu được đồng bộ hóa với cùng tín hiệu đồng hồ (được cung cấp bởi thiết bị master thông qua chân SCK).

**Slave select (NSS) pin management**

Để tiết kiệm chân chip, phần cứng hỗ trợ việc quản lý chân slave select bằng phần mềm. Quản lý lựa chọn slave phần cứng hoặc phần mềm có thể được đặt bằng thanh ghi.

• Quản lý NSS phần mềm

Thông tin chọn chip slave được điều khiển bên trong bởi giá trị của bit SSI trong thanh ghi SPI\_CR1. Chân NSS bên ngoài vẫn nhàn rối cho các ứng dụng khác.

• Quản lý NSS phần cứng

Hai cấu hình có thể tùy thuộc vào cấu hình đầu ra NSS

- Đã bật đầu ra NSS

Cấu hình này chỉ được sử dụng khi thiết bị hoạt động ở chế độ SPI. Tín hiệu NSS được điều khiển ở mức thấp khi chủ bắt đầu liên lạc và được giữ ở mức thấp cho đến khi SPI bị vô hiệu hóa.

- Ngắt đầu ra NSS

Cấu hình này cho phép nhiều thiết bị master hoạt động ở chế độ SPI. Đối với các thiết bị được đặt làm slave, chân NSS hoạt động như một đầu vào NSS cổ điển: slave được chọn khi NSS thấp và không được chọn khi NSS cao.

**Clock phase and clock polarity**

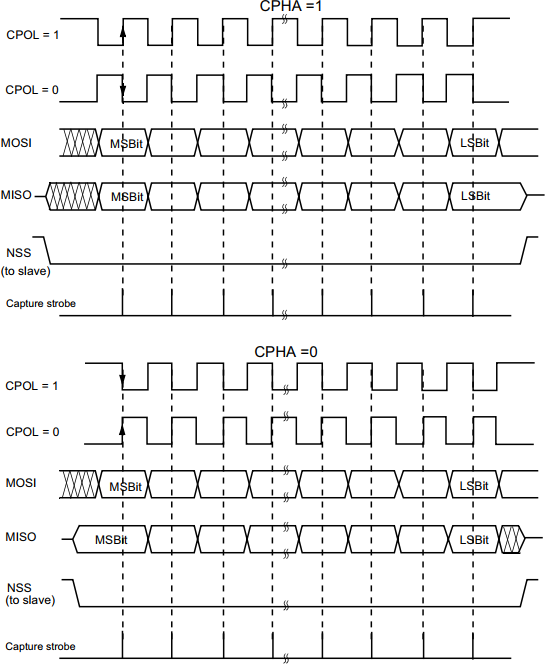
Hai thống số này là hai thông số quan trọng quyets định thời điểm chuyển trạng thái của xung đồng hồ và thời điểm lấy mẫu tín hiệu. Mỗi tham số có hai giá trị nên ta có thể có 4 kịch bản timing cho SPI/I2S.

Clock polarity(CPOL): tham số này quyết định trạng thái tích cực của xung đồng hồ và thời điểm bắt đầu chu kỳ.

* + CPOL = 0: Clock tích cực thấp và sườn lên xung đồng hồ đầu tiên là thời điểm bawets đầu truyền dữ liệu
  + CPOL = 1: Clock tích cực cao và sườn xuống xung đồng hồ đầu tiên là thời điểm bắt đầu truyền dữ liệu

Clock Phase(CPHA): Tham số này quyết định thời điểm lấy mẫu tín hiệu.

* + CPHA = 0: Lấy mẫu tại sườn đầu tiên sau sườn leading
  + CPHA = 1: Lấy mẫu tại sườn thứ hai sau sườn leading



*Figure 17 timing diagram của SPI*

**Định dạng khung truyền**

Phần cứng cho phép lựa chọn chế độ dịch chuyển dữ liệu theo bít trọng số cao trước(MSB) hay bit trong số thấp trước(LSB). Điều này được thực hiện bởi việc ghi vào một thanh ghi điều khiển của SPI.

Độ dài mỗi khung truyền (8 hay 16) bít được lập trình phần mềm bằng cách cấu hình thanh ghi điều khiển của SPI. Cấu hình này phải thống nhất về hai phía master và slave.

**Cờ ngắt**

Với STM32F4xx, có một số cờ để quản lý các hoạt động của giao thức SPI.

Cờ trống bộ đệm Tx (TXE)

Khi được dựng, cờ này biểu thị rằng bộ đệm Tx trống và dữ liệu tiếp theo được truyền có thể được tải vào bộ đệm. Cờ TXE bị xóa khi ghi vào thanh ghi dữ liệu của SPI.

Bộ đệm Rx không trống (RXNE)

Khi được dựng, cờ này biểu thị rằng có dữ liệu nhận được hợp lệ trong bộ đệm Rx. Nó bị xóa khi thanh ghi dữ liệu SPI được đọc.

Cờ báo bận

Cờ BSY này được đặt và xóa bằng phần cứng (ghi vào cờ này không có hiệu lực). Cờ BSY chỉ trạng thái của lớp truyền thông của SPI.

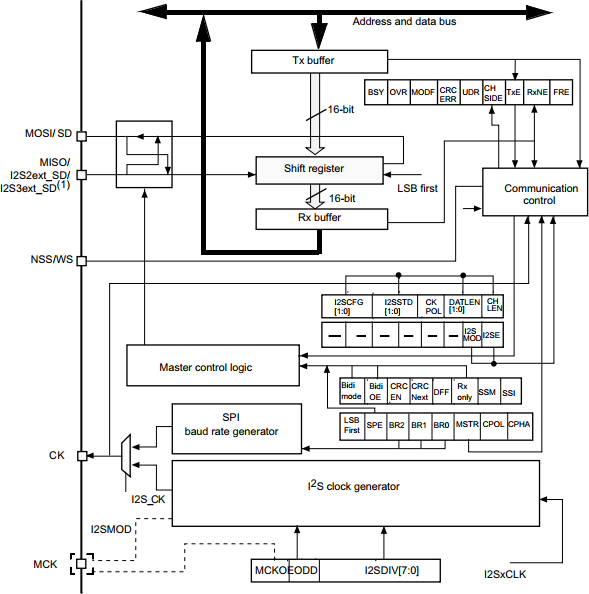
Khi BSY được đặt, nó cho biết SPI đang bận giao tiếp. Có một ngoại lệ trong chế độ nhận chính / chế độ nhận hai chiều trong đó cờ BSY được giữ ở mức thấp trong khi tiếp nhận.

Cờ BSY rất hữu ích để phát hiện kết thúc chuyển dữ liệu nếu phần mềm muốn tắt SPI và vào chế độ Dừng (hoặc tắt đồng hồ ngoại vi). Điều này tránh làm hỏng việc chuyển dữ liệu cuối cùng.

Thủ tục truyền và nhận dữ liệu

Khi tiếp nhận, dữ liệu được nhận và sau đó được lưu trữ vào bộ đệm Rx bên trong, khi truyền, dữ liệu trước tiên được lưu trữ vào bộ đệm Tx bên trong trước khi được truyền. Truy cập đọc của thanh ghi dữ liệu trả về giá trị bộ đệm Rx trong khi truy cập ghi vào dữ liệu lưu dữ liệu vào bộ đệm Tx.

### **2.4.2 Chức năng khối I2S**



*Ảnh 18 Sơ đồ khối I2S*

I2S chia sẻ ba chân chung với SPI:

• SD: Dữ liệu nối tiếp (được ánh xạ trên chân MOSI) để truyền hoặc nhận hai kênh dữ liệu đa kênh (chỉ ở chế độ bán song công).

• WS: Word Chọn (được ánh xạ trên chân NSS) là đầu ra tín hiệu điều khiển dữ liệu ở chế độ master và đầu vào ở chế độ slave.

• CK: Đồng hồ nối tiếp (được ánh xạ trên chân SCK) là đầu ra đồng hồ nối tiếp ở chế độ master và đầu vào đồng hồ nối tiếp ở chế độ slave.

• I2S2ext\_SD và I2S3ext\_SD: các chân bổ sung (được ánh xạ trên chân MISO) để điều khiển chế độ song công hoàn toàn của I2S.

Một pin bổ sung có thể được sử dụng khi cần một đầu ra đồng hồ chính cho một số thiết bị âm thanh bên ngoài:

• MCK: xung đồng hồ (được ánh xạ riêng) được sử dụng, khi I2S được định cấu hình ở chế độ master để xuất xung, xung đồng hồ bổ sung này được tạo ở tốc độ tần số được cấu hình sẵn bằng 256 × FS, trong đó FS là tần số lấy mẫu âm thanh.

I2S sử dụng trình tạo xung đồng hồ riêng để tạo xung đồng hồ đồng bộ khi được đặt ở chế độ chính. Bộ tạo xung đồng hồ này cũng là nguồn của đầu ra xung MCK. Hai thanh ghi bổ sung có sẵn trong chế độ I2S. Một cái được liên kết với cấu hình bộ tạo xung nhịp và cái còn lại là thanh ghi cấu hình I2S chung.

**Các giao thức âm thanh hỗ trợ**

Các bus chỉ phải xử lý dữ liệu âm thanh thường được ghép theo thời gian trên hai kênh: kênh bên phải và kênh bên trái. Tuy nhiên, chỉ có một thanh ghi 16 bit cho việc truyền và nhận. Vì vậy, tùy thuộc vào phần mềm ghi vào thanh ghi dữ liệu giá trị tương ứng với phía kênh được xem xét hoặc để đọc dữ liệu từ thanh ghi dữ liệu và xác định kênh tương ứng bằng cách kiểm tra thanh ghi trạng thái của SPI. Kênh trái luôn được gửi trước theo sau là kênh phải. Dữ liệu có thể được gửi với định dạng:

• Dữ liệu 16 bit được đóng gói trong khung 16 bit

• Dữ liệu 16 bit được đóng gói trong khung 32 bit

• Dữ liệu 24 bit được đóng gói trong khung 32 bit

• Dữ liệu 32 bit được đóng gói trong khung 32 bit

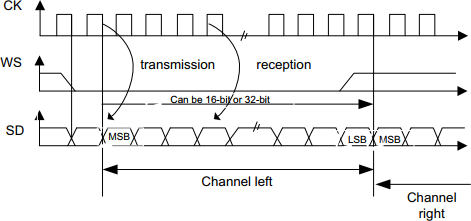
Khi sử dụng dữ liệu 16 bit được mở rộng trên gói 32 bit, 16 bit đầu tiên (MSB) là các bit MSB, LSB 16 bit bị set về 0 mà không cần bất kỳ hành động phần mềm hoặc yêu cầu DMA nào ).

Khung dữ liệu 24 bit và 32 bit cần hai hoạt động đọc hoặc ghi CPU đến / từ thanh ghi dữ liệu hoặc hai trigger DMA nếu DMA được ưu tiên cho ứng dụng. Đối với khung dữ liệu 24 bit cụ thể, 8 bit LSB được mở rộng thành 32 bit với 0 bit (bằng phần cứng).

Đối với tất cả các định dạng dữ liệu và tiêu chuẩn truyền thông, bit quan trọng nhất luôn được gửi trước (MSB trước).

Giao diện I2S hỗ trợ bốn tiêu chuẩn âm thanh. Tôi sẽ mô tả chi tiết giao thức nào tôi sử dụng trong ứng dụng này, giao thức âm thanh Phillip.

Đối với tiêu chuẩn này, tín hiệu WS được sử dụng để chỉ ra kênh nào đang được truyền. Nó được kích hoạt một chu kỳ xung nhịp CK trước khi bit đầu tiên (MSB) được truyền.



*Figure 19 Giao thức I2S Phillips*

Dữ liệu được lấy mẫu trên sườn xuống của CK (đối với máy phát) và được đọc trên sườn lên (đối với máy thu). Tín hiệu WS cũng được lấy mẫu ở sườn xuống của CK.

Để truyền, mỗi lần bit MSB được ghi vào thanh ghi dữ liệu, cờ TXE được đặt và ngắt của nó, nếu được phép, được tạo để tải thanh ghi dữ liệu với giá trị mới cần gửi.

Để nhận, cờ RXNE được đặt và ngắt của nó, nếu được phép, được tạo khi nhận được nửa từ bit MSB đầu tiên.

**Cờ và quản lý ngắt trong modun I2S**

Để quản lý các hoạt động truyền và nhận, giao thức I2S hỗ trợ các cờ ngắt bên dưới:

Cờ báo bận (BSY)

Cờ BSY được đặt và xóa bằng phần cứng (ghi vào cờ này không có hiệu lực). Nó chỉ ra trạng thái của giao tiếp của I2S.

Khi BSY được đặt, nó chỉ ra rằng I2S đang bận giao tiếp. Có một ngoại lệ trong chế độ nhận chính trong đó cờ BSY được giữ ở mức thấp trong quá trình nhận.

Cờ BSY rất hữu ích để phát hiện kết thúc chuyển dữ liệu nếu phần mềm cần vô hiệu hóa I2S.

Điều này tránh làm hỏng việc chuyển dữ liệu cuối cùng. Đối với điều này, các thủ tục được mô tả dưới đây phải được tôn trọng nghiêm ngặt.

Cờ BSY được đặt khi quá trình truyền bắt đầu, ngoại trừ khi I2S ở chế độ máy thu chính. Cờ BSY bị xóa:

• Khi quá trình truyền hoàn tất (ngoại trừ ở chế độ truyền chính, trong đó giao tiếp được coi là liên tục)

• Khi I2S bị vô hiệu hóa

Khi truyền nhận liên tục:

• Ở chế độ truyền chính, cờ BSY được giữ ở mức cao trong tất cả các lần chuyển

• Trong chế độ phụ thuộc, cờ BSY xuống thấp trong một chu kỳ đồng hồ I2S giữa mỗi lần chuyển

Cờ trống bộ đệm TX (TXE)

Khi được đặt, cờ này cho biết bộ đệm Tx trống và dữ liệu tiếp theo được truyền có thể được tải vào nó. Cờ TXE được xóa khi bộ đệm Tx đã chứa dữ liệu được truyền. Nó cũng được xóa khi I2S bị tắt.

Bộ đệm RX không trống (RXNE)

Khi được đặt, cờ này biểu thị rằng có dữ liệu nhận được hợp lệ trong Bộ đệm RX. Nó được xóa lại khi thanh ghi dữ liệu được đọc.

Cờ báo kênh

Trong chế độ truyền, cờ này được làm mới khi TXE dựng. Nó chỉ ra phía kênh mà dữ liệu cần truyền. Trong trường hợp xảy ra sự cố lỗi trong chế độ truyền phụ thuộc, cờ này không đáng tin cậy và I2S cần được tắt và bật trước khi tiếp tục truyền nhận.

Trong chế độ nhận, cờ này được làm mới khi dữ liệu được nhận vào thanh ghi dliệu. Nó

cho biết dữ liệu phía kênh nào đã được nhận. Lưu ý rằng trong trường hợp có lỗi (như Overun), cờ này trở nên vô nghĩa và I2S nên được khởi động lại bằng cách vô hiệu hóa và sau đó bật nó (với cấu hình nếu cần thay đổi).

## Giao Thức I2C

## Giới thiệu giao thức

I²C là một bus máy tính nối tiếp, nhiều master, nhiều slave, chuyển mạch gói, kết thúc đơn, được phát minh vào năm 1982 bởi Philips Semiconductor (nay là NXP Semiconductor). Nó được sử dụng rộng rãi để gắn IC ngoại vi tốc độ thấp hơn vào bộ xử lý và vi điều khiển trong giao tiếp nội bộ, khoảng cách ngắn.

## Các tính năng I2C

Trong điện tử tiêu dùng, viễn thông và điện tử công nghiệp, thường có nhiều điểm tương đồng giữa các thiết kế dường như không liên quan. Ví dụ: gần như mọi hệ thống bao gồm:

• Một số thiết bị điều khiển thông minh, thường là vi điều khiển

• Các mạch đa năng như trình điều khiển LCD và LED, cổng I/O từ xa, RAM, EEPROM, đồng hồ thời gian thực hoặc bộ chuyển đổi A/D và D/A

• Các mạch định hướng ứng dụng như mạch điều chỉnh kỹ thuật số và xử lý tín hiệu cho hệ thống radio và video, cảm biến nhiệt độ và thẻ thông minh

Bus này được gọi là Inter IC hoặc I2C-bus. Tất cả các thiết bị tương thích với bus I2C đều tích hợp giao diện trên chip cho phép chúng giao tiếp trực tiếp với nhau thông qua bus I2C. Khái niệm thiết kế này giải quyết nhiều vấn đề giao thức gặp phải khi thiết kế mạch điều khiển kỹ thuật số.

Dưới đây là một số tính năng của I2C-bus:

• Chỉ có hai dây được yêu cầu; một dòng dữ liệu nối tiếp (SDA) và một dòng đồng hồ nối tiếp (SCL).

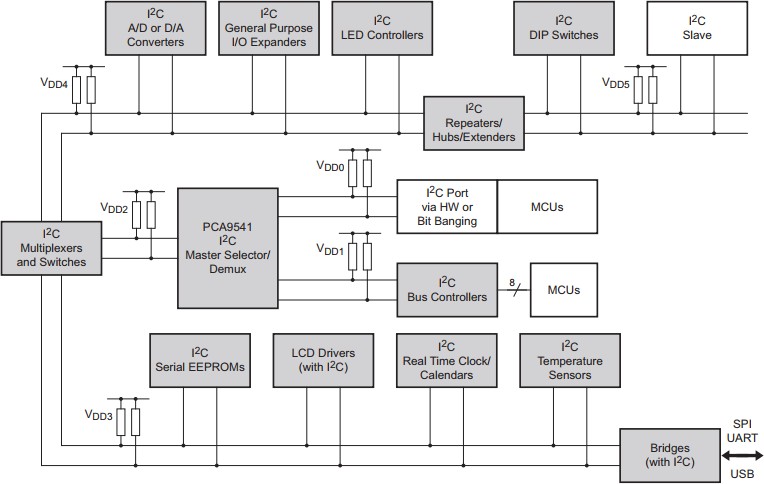
• Mỗi thiết bị được kết nối với bus được đánh địa chỉ bằng phần mềm theo một địa chỉ duy nhất và luôn tồn tại các mối quan hệ master / slave đơn giản.

• Đây là một bus hỗ trợ chế độ nhiều master thực sự bao gồm phát hiện xung đột và gải quyết xung đột để ngăn ngừa mất mát dữ liệu nếu hai hoặc nhiều master cùng lúc bắt đầu truyền dữ liệu.

• Truyền dữ liệu hai chiều, định dạng 8 bit, có thể được thực hiện với tốc độ lên tới 100 kbit / giây ở chế độ Tiêu chuẩn, lên tới 400 kbit / giây ở chế độ Nhanh, lên tới 1 Mbit / giây ở chế độ Nhanh hoặc lên đến 3,4 Mbit / s ở chế độ Tốc độ cao.

• Truyền dữ liệu một chiều, 8 bit, định hướng, truyền lên tới 5 Mbit / s ở chế độ Siêu nhanh

• Bộ lọc trên chip loại bỏ các xung đột trên đường dữ liệu bus để bảo vệ tính toàn vẹn dữ liệu.



*Figure 20 Ví dụ về ứng dụng của I2C bus*

## Các chế độ hoạt động I2C

Ban đầu, bus I2C bị giới hạn hoạt động ở tốc độ 100 kbit/s. Theo thời gian đã có một số bổ sung cho đặc điểm kỹ thuật để bây giờ có năm loại tốc độ hoạt động. Các thiết bị ở chế độ Chuẩn, Chế độ nhanh (Fm), Chế độ nhanh Plus (Fm +) và Chế độ tốc độ cao (Chế độ Hs) đều tương thích với mọi thiết bị có thể được vận hành ở tốc độ bus thấp hơn. Các thiết bị chế độ siêu nhanh không tương thích với các phiên bản trước.

• Bus hai chiều:

- Chế độ tiêu chuẩn (Sm), với tốc độ bit lên tới 100 kbit / s

- Chế độ nhanh (Fm), với tốc độ bit lên tới 400 kbit / s

- Chế độ nhanh Plus (Fm +), với tốc độ bit lên tới 1 Mbit / s

- Chế độ tốc độ cao (chế độ Hs), với tốc độ bit lên tới 3,4 Mbit / s.

• Bus một chiều:

- Chế độ cực nhanh (UFm), với tốc độ bit lên tới 5 Mbit / s

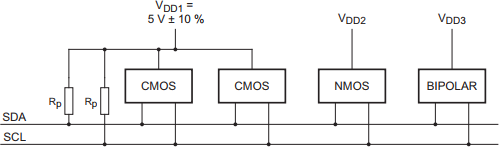
## Giao thức I2C-bus

Trong phần bên dưới của đồ án này, tôi chỉ mô tả hoạt động giao thức I2C cho các chế độ bus hai chiều I2C.

Tín hiệu SDA và SCL

Giao thức bus I2C sử dụng hai dây để truyền và nhận: Dữ liệu nối tiếp (SDA) và Đồng hồ nối tiếp (SCL). Cả SDA và SCL đều là các đường hai chiều, được kết nối với điện áp nguồn thông qua một điện trở nguồn hiện tại hoặc kéo lên. Khi I2C bus nhàn rỗi, cả hai dòng đều ở mức cao. Các giai đoạn đầu ra của các thiết bị được kết nối với bus phải có cống mở hoặc bộ thu mở để thực hiện chức năng AND. Dữ liệu trên bus I2C có thể được truyền với tốc độ lên tới 100 kbit / giây ở chế độ Tiêu chuẩn, lên tới 400 kbit / giây ở chế độ Nhanh, tối đa 1 Mbit / giây, ở chế độ Nhanh đến 3,4 Mbit / s, ở chế độ Tốc độ cao. Điện dung của bus giới hạn số lượng thiết bị được kết nối với bus.

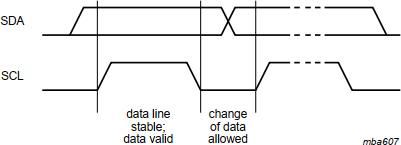
Đối với một ứng dụng chính duy nhất, đầu ra Master SCL có thể là một thiết kế trình điều khiển kéo đẩy nếu không có thiết bị nào trên I2C bus sẽ kéo dài xung đồng hồ.



*Ảnh 21 Các thiết bị chia sẻ một đường bus*

**Tính hợp lệ của dữ liệu:**

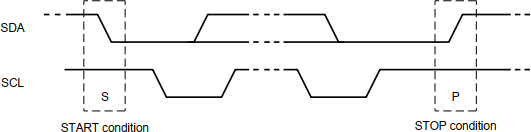
Dữ liệu trên dòng SDA phải ổn định trong khoảng thời gian xung đồng hồ ở mức cao. Trạng thái cao hoặc thấp của dòng dữ liệu chỉ có thể thay đổi khi tín hiệu xung đồng hồ trên dòng SCL ở mức thấp. Một xung đồng hồ được tạo cho mỗi bit dữ liệu được truyền.



*Figure 22 Các bit truyền trên I2C-bus*

**Tín hiệu bắt đầu và kết thúc**

All transactions begin with a START (S) and are terminated by a STOP (P). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.



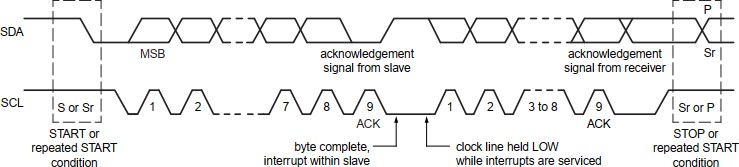
*Figure 23 START và STOP bit trên I2C bus*

Điều kiện bắt đầu và kết thúc luôn được tạo bởi master. I2C bus được coi là bận sau điều kiện bắt đầu. I2C bus được coi là nhàn rỗi một lần nữa sau điều kiện kết thúc.

I2C bus vẫn bận nếu điều kiện bắt đầu lặp lại (Sr) được tạo thay vì điều kiện kết thúc. Về mặt này, các điều kiện bắt đầu (S) và bắt đầu lại (Sr) lặp lại giống nhau về mặt chức năng.

## Định dạng byte

Mỗi byte đặt trên dòng SDA phải dài tám bit. Số lượng byte có thể được truyền mỗi lần truyền không bị hạn chế. Mỗi byte phải được theo sau bởi một bit ACK. Dữ liệu được truyền bằng Bit trọng số cao (MSB) trước tiên. Nếu một slave không thể nhận hoặc truyền một byte dữ liệu hoàn chỉnh khác cho đến khi nó thực hiện một số chức năng khác, ví dụ như phục vụ ngắt nội bộ, nó có thể giữ dòng xung đồng hồ SCL thấp để buộc master vào trạng thái chờ. Truyền dữ liệu sau đó tiếp tục khi slave sẵn sàng cho một byte dữ liệu khác và giải phóng dòng xung đồng hồ SCL.



*Figure 24 Dữ liệu truyền trên I2C-bus*

## Tín hiệu ACK và NACK

Việc xác nhận diễn ra sau mỗi byte được truyền. Bit ACK cho phép slave báo hiệu cho master rằng byte đã được nhận thành công và một byte khác có thể được gửi. Master tạo ra tất cả các xung đồng hồ, bao gồm cả xung đồng hồ thứ chín cho tín hiệu ACK.

Tín hiệu ACK được định nghĩa như sau: master giải phóng đường SDA trong xung đồng hồ cho tín hiệu ACK để slave có thể kéo đường SDA xuống mức thấp và nó vẫn ổn định ở mức thấp trong khoảng thời gian ở mức cao của xung đồng hồ này.

Khi SDA duy trì trạng thái cao trong xung nhịp thứ chín này, tín hiệu này được xác định là tín hiệu NACK. Sau đó, master có thể tạo một điều kiện dừng để hủy bỏ chuyển hoặc điều kiện bắt đầu lặp lại để bắt đầu chuyển mới. Có năm điều kiện dẫn đến việc tạo ra một NACK:

1. Không có slave nào có mặt trên bus với địa chỉ được truyền nên không có thiết bị nào phản hồi với ACK.

2. Slave không thể nhận hoặc truyền vì nó đang thực hiện một số chức năng thời gian thực và chưa sẵn sàng để bắt đầu liên lạc với master.

3. Trong quá trình chuyển, slave nhận được dữ liệu hoặc lệnh mà nó không hiểu.

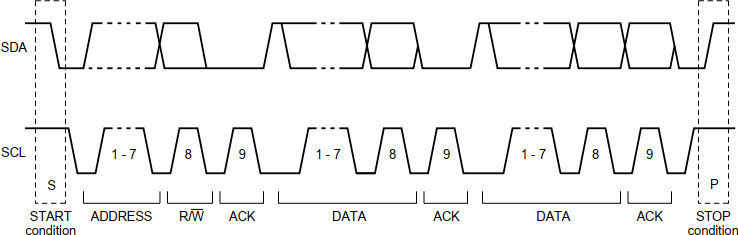
4. Trong quá trình truyền, slave không thể nhận thêm byte dữ liệu.

5. Một Slave truyền phải báo hiệu kết thúc chuyển đến master slave.

## Địa chỉ slave và R/W bit

Truyền dữ liệu theo định dạng được hiển thị trong hình dưới đây. Sau điều kiện bắt đầu (S), một địa chỉ slave được gửi. Địa chỉ này dài bảy bit, theo sau là bit thứ tám, là bit hướng dữ liệu (R / W) - một bit 0, biểu thị một đường truyền (WRITE), một bit 1 chỉ ra yêu cầu dữ liệu (READ). Truyền dữ liệu luôn được kết thúc bởi một điều kiện STOP (P) được tạo bởi master.

Tuy nhiên, nếu master vẫn muốn giao tiếp trên bus, nó có thể tạo điều kiện bắt đầu lặp lại (Sr) và xử lý một slave khác mà không cần tạo điều kiện kết thức trước. Sự kết hợp khác nhau của các yêu cầu đọc / ghi sau đó có thể có trong một lần chuyển như vậy.



*Figure 25 Một khung truyền hoàn thiện*

## START byte

Vi điều khiển có thể được kết nối với bus I2C theo hai cách. Một bộ vi điều khiển có giao diện I2C-bus phần cứng trên chip có thể được lập trình để chỉ bị gián đoạn bởi các yêu cầu từ bus. Khi thiết bị không có giao diện như vậy, nó phải liên tục theo dõi bus thông qua phần mềm. Rõ ràng, bộ vi điều khiển giám sát hoặc thăm dò bus càng nhiều lần thì càng mất ít thời gian để thực hiện chức năng dự định của nó.

Do đó, có sự khác biệt về tốc độ giữa các thiết bị phần cứng nhanh và vi điều khiển tương đối chậm, phụ thuộc vào phần mềm.

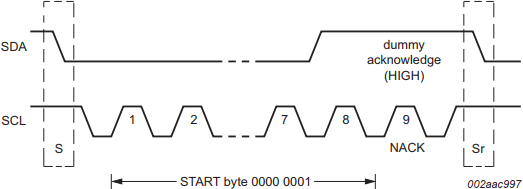
Trong trường hợp này, việc truyền dữ liệu có thể được thực hiện trước một thủ tục bắt đầu dài hơn bình thường. Thủ tục bắt đầu bao gồm:

• Điều kiện bắt đầu (S).

• byte bắt đầu (0000 0001).

• Một xung nhịp xác nhận (ACK).

• Một điều kiện bắt đầu lặp lại (Sr).



*Figure 26 Byte bắt đầu*

Sau khi điều kiện bắt đầu được truyền bởi một master yêu cầu truy cập bus, byte bắt đầu (0000 0001) được truyền đi. Do đó, một bộ vi điều khiển khác có thể lấy mẫu dòng SDA ở tốc độ lấy mẫu thấp cho đến khi một trong bảy số không trong byte bắt đầu được phát hiện. Sau khi phát hiện mức thấp này trên dòng SDA, vi điều khiển có thể chuyển sang tốc độ lấy mẫu cao hơn để tìm điều kiện bắt đầu lặp lại Sr sau đó được sử dụng để đồng bộ hóa.

Một bộ thu phần cứng đặt lại khi nhận được điều kiện bắt đầu lặp lại Sr và do đó bỏ qua byte bắt đầu.

Xung đồng hồ liên quan đến xác nhận được tạo sau byte bắt đầu. Điều này chỉ hiện diện để phù hợp với định dạng xử lý byte được sử dụng trên bus. Không có thiết bị nào được phép thừa nhận byte bắt đầu.

## Giao thức USB

USB (Universal Serial Bus Interface) hiện được thiết lập tốt như một giao thức để kết nối với máy tính. Trong nhiều lĩnh vực, nó đã vượt qua hoàn toàn RS232 và giao thức song song hoặc Centrics cho máy in, và nó cũng được sử dụng rộng rãi cho thẻ nhớ, chuột máy tính, bàn phím và cho nhiều chức năng khác. Một trong những lợi thế của USB là tính linh hoạt của nó. Một điều nữa là tốc độ mà USB cung cấp.

USB cho phương thức truyền dữ liệu nối tiếp khá nhanh để truyền dữ liệu, tuy nhiên cũng có thể lấy nguồn thông qua các đầu nối và điều này đã tăng thêm sự phổ biến của USB khi nhiều phụ kiện máy tính có công suất thấp. USB được tìm thấy một tiện ích rộng rãi từ bộ nhớ đến ổ đĩa. Giao thức USB phát triển là kết quả của nhu cầu về giao thức truyền dữ liệu dễ sử dụng và giao thức hỗ trợ tốc độ dữ liệu cao hơn, đây là yêu cầu chính đối với ngành công nghiệp máy tính và thiết bị ngoại vi.



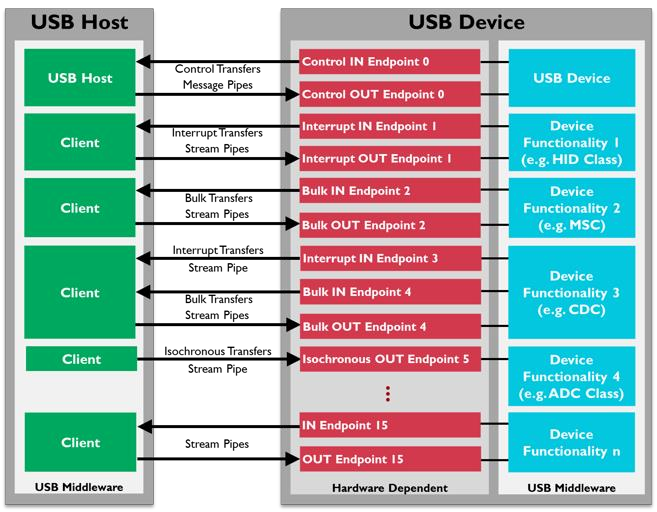
*Figure 27 Thiết bị lưu trữ USB*

Với USB 1.0 được thiết lập tốt, tốc độ truyền dữ liệu nhanh hơn được yêu cầu và do đó, một đặc điểm kỹ thuật mới, USB 2.0 đã được phát hành. Với tầm quan trọng của USB đã được thiết lập, không mất nhiều thời gian để chuẩn mới được thông qua.

USB định hình vị trí của nó trên thị trường máy tính, các cải tiến khác của tiêu chuẩn đã được nghiên cứu. Với nhu cầu về tính di động trong nhiều lĩnh vực của ngành công nghiệp điện tử, bước đi dự đoán tiếp theo cho USB có thể là sử dụng giao thức không dây. Để làm cho ddieuf đó khả thi, USB sẽ cần phải sử dụng một phương pháp linh hoạt đã chứng minh sự thành công cho giao thức có dây. Hơn nữa, giao thức USB không dây phải có khả năng gửi dữ liệu ở tốc độ cao hơn so với kết nối USB 2 có dây.

## Kết nối USB

USB là một bus nối tiếp, trong đó tất cả việc truyền và nhận dữ liệu được bắt đầu bởi host USB. Dữ liệu được truyền đến hoặc từ các điểm cuối trong thiết bị USB. Máy khách trong máy chủ lưu trữ USB lưu trữ dữ liệu trong bộ đệm, nhưng không có điểm cuối. Như được hiển thị bên dưới các lớp truyền dữ liệu khác nhau có thể được nhìn thấy. Sự tương tác giữa các lớp khác nhau là kết nối máy chủ-thiết bị logic giữa mỗi lớp ngang. Giữa các dữ liệu kết nối hợp lý được chuyển bằng cách sử dụng đường ống.



*Figure 28 Kết nối logic giữa USB client và host*

## Ưu nhược điểm của kết nối USB

## USB có nhiều ưu điểm khi so sánh với các công nghệ khác, nhưng nó cũng có một số nhược điểm cần được xem xét khi quyết định sử dụng công nghệ.

*Table 2 Ưu nhược điểm của giao thức USB*

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Thân thiện với người dùng | Tốc độ truyền dữ liệu không nhanh như các giao thức khác |
| Nhiều ứng dụng có thể tìm thấy tiện ích của nó với tốc độ truyền dữ liệu trong phạm vi. | Sức chứ và tốc độ giới hạn |
| Kết nối trơn tru |  |
| Nhiều loại cổng kết nối |  |
| Chi phí thấp |  |

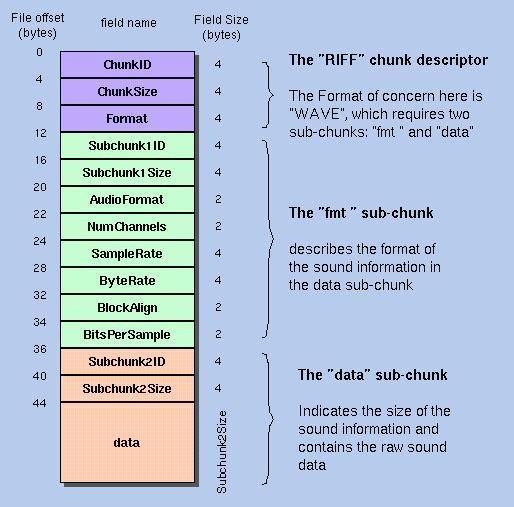
USB có nhiều ưu điểm và đây là lý do tại sao nó được sử dụng rộng rãi. Tuy nhiên, tính đơn giản và dễ sử dụng của nó, có nghĩa là không phải lúc nào cũng có thể áp dụng trong các ứng dụng đòi hỏi giao diện phức tạp hơn để truyền dữ liệu tốc độ rất cao.

## Định dạng file âm thanh WAV

## Tổng quan về định dạng WAV

Định dạng tệp Wave là định dạng tệp cục bộ của Windows để lưu trữ thông tin âm thanh kỹ thuật số trên máy vi tính. Nó đã trở nên nổi bật trong số các định dạng âm thanh được vi tính hóa rộng rãi nhất trên PC vì sự nổi tiếng của Windows và số lượng lớn các dự án được sáng tác cho sân khấu. Trong định dạng tệp này, việc sắp xếp dữ liệu thực hiện theo dang little endian hay cụ thể byte ít quan trọng nhất phải xuất hiện đầu tiên.

Các tệp WAV sử dụng cấu trúc định dạng tệp trao đổi tài nguyên chung để lưu trữ nội dung của tệp vào các khối khác nhau. Phương pháp tổ chức này làm cho một chương trình trở nên mạnh mẽ bằng cách cho phép nó bỏ qua các loại khối không thể nhận ra và tiếp tục xử lý các khối dữ liệu đã biết. Hình dưới đây cho thấy bố cục tệp WAV cơ bản.



*Figure 29 Định dạng WAV*

## Wave file header

Tiêu đề file WAV tuân theo cấu trúc RIFF tiêu chuẩn. 8 byte đầu tiên của tệp là một tiêu đề RIFF có ID của RIFF và kích thước chunk giống như kích thước tệp trừ 8 byte cho phần header. Các tệp WAV được lưu trữ không nén do đó chúng có thể khá lớn, nhưng chúng không thể vượt quá 4 gigabyte do thực tế là trường tiêu đề kích thước tệp là một số nguyên không dấu 32 bit.

Một tệp WAVE thường chỉ là một tệp RIFF với một đoạn "WAVE" duy nhất bao gồm hai phần con - một đoạn "fmt" chỉ định định dạng dữ liệu và một đoạn "dữ liệu" chứa dữ liệu mẫu thực tế.

Bảng sau đây cung cấp chi tiết về tiêu đề tệp WAV.

.

*Table 3 Wave File Header*

|  |  |  |  |
| --- | --- | --- | --- |
| Số Byte | Kích cỡ | Miêu tả | Giá trị |
| 0-3 | 4 | Chunk ID | “RIFF” (0x52494646) |
| 4-7 | 4 | Chunk Data Size | 8 (file size) |
| 8-11 | 4 | RIFF type | “WAVE”  (0x57415645) |

## Định dạng Chunk

Định dạng chunk có thông tin chi tiết về việc lưu trữ dạng WAV dữ liệu và phát nó bao gồm chế độ nén, số kênh, tốc độ lấy mẫu, số bit trong mỗi mẫu và các thuộc tính khác.

## Kênh

Số lượng kênh rất quan trọng: một kênh là Mono, hai kênh là stereo - có các sóng khác nhau cho loa trái và phải. Âm thanh vòm 5.1 có 5 kênh, một trong số đó là âm thanh thấp nhất và thường được gửi đến loa siêu trầm. Một lần nữa, mỗi kênh chứa dữ liệu âm thanh độc lập với tất cả các kênh khác, mặc dù tất cả các kênh sẽ có cùng độ dài.

## Khung truyền

- Khung giống như một mẫu, nhưng ở định dạng đa kênh - đó là ảnh chụp nhanh tất cả các kênh tại một điểm dữ liệu cụ thể.p

## Tần số lấy mẫu

Số lượng mẫu (hoặc khung) tồn tại cho mỗi giây dữ liệu. Trường này được biểu thị bằng Hz. Ví dụ, âm thanh chất lượng CD có 44.100 mẫu mỗi giây. Tỷ lệ lấy mẫu cao hơn có nghĩa là âm thanh độ trung thực cao hơn.

## Độ dài từ mẫu

Số lượng bit có sẵn cho một mẫu. Độ sâu bit phổ biến là 8 bit, 16 bit và 32 bit. Một mẫu hầu như luôn được đại diện bởi một loại dữ liệu nguyên gốc, chẳng hạn như byte, short hoặc int. Độ sâu bit cao hơn có nghĩa là mỗi mẫu có thể chính xác hơn, dẫn đến âm thanh có độ trung thực cao hơn.

## Data chunk

*Table 4 Wav Data chunk*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BYTE NUMBER | SIZE (Bytes) | DESCRIPTION | VALUES | |
| 0-3 | 4 | Chunk ID | "fmt " (0x666D7420) |  |
|  | |
| 4-7 | 4 | Kích cỡ Chunk | Độ dài định dạng | |
| Chunk (always 0x10) | |
| 8-9 | 2 | Mã nén | Luôn luôn là 0x01 | |
| 10-11 | 2 | Thứ tự kênh | 0x01=mono 0x02=stereo | |

|  |  |  |  |
| --- | --- | --- | --- |
| 12-15 | 4 | Tần số lấy mẫu | Nhị phân tính bằng Hertz |
| 16-19 | 4 | Số Byte truyền trên một giây |  |
| 20-21 | 2 | Số byte trên mẫu | 1=8 bit mono, 2=8 bit  stereo, 3=16 bit  mono, 4= 16 bit stereo |
| 22-23 | 2 | Số bit trên mẫu |  |

# CHương III: Thiết bị phần cứng và công cụ phát triển

Trong chương này, tôi sẽ giới thiệu các công cụ phát triển phần cứng và phần mềm tôi đã sử dụng trong dự án của mình bao gồm:

* + STM32F4 Discovery kit
  + KEIL C uVision 5
  + STM 32 Cube Mx

## STM32F4 Discovery kit

Kit STM32F4DISCOVERY là bộ công cụ phát triển chi phí thấp và dễ sử dụng để nhanh chóng bắt đầu phát triển với bộ vi điều khiển hiệu suất cao STM32F407VG.

## Tính năng

STM32F4DISCOVERY cung cấp các tính năng sau:

• Vi điều khiển STM32F407VGT6 có ARM Cortex M4 32 bit với lõi FPU, bộ nhớ Flash 1 Mbyte, RAM 192 Kbyte trong gói LQFP100

• Mạch nạp ST-LINK / V2 tích hợp cùng bo mạch trên STM32F4DISCOVERY hoặc ST-LINK / V2-A trên STM32F407G-DISC1

• USB ST-LINK với khả năng liệt kê ba giao diện khác nhau:

- Cổng COM ảo (chỉ có ST-LINK / V2-A)

- Lưu trữ lớn (chỉ với ST-LINK / V2-A)

- Cổng debug

• Kit được cấp nguồn qua:

- Qua cổng USB

- Nguồn điện bên ngoài: 3 V và 5 V

• Gia tốc kế 3 trục LIS302DL hoặc LIS3DSH ST MEMS

• Cảm biến âm thanh MP45DT02 ST MEMS micrô kỹ thuật số đa hướng

• Bộ xử lý âm thanh CS43L22 với trình điều khiển loa lớp D tích hợp

• Tám đèn LED:

- LD1 (đỏ / xanh) cho giao tiếp USB

- LD2 (màu đỏ) khi bật nguồn 3,3 V

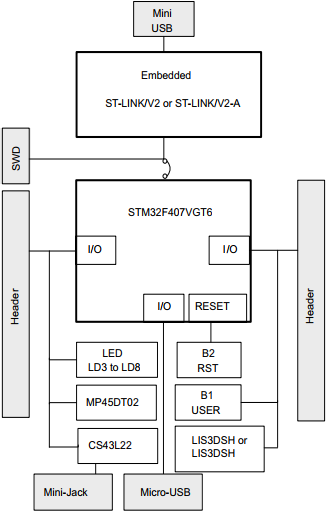
- Bốn đèn LED người dùng, LD3 (cam), LD4 (xanh lá cây), LD5 (đỏ) và LD6 (xanh dương)

- 2 đèn LED USB OTG LD7 (xanh) VBUS và LD8 (đỏ) quá dòng

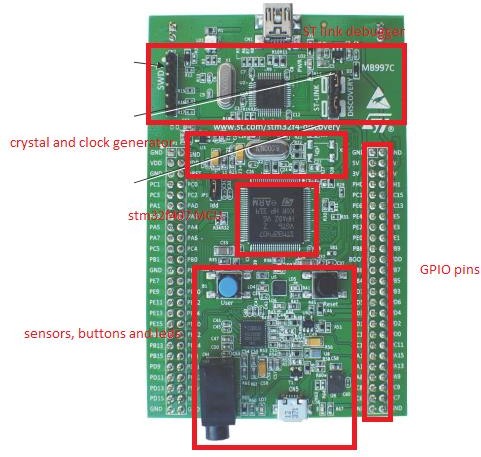
• Hai nút ấn (người dùng và reset)

• USB OTG FS với đầu nối micro-AB

## Cấu trúc phần cứng



*Figure 30 Sơ đồ khối kit STM32F4*



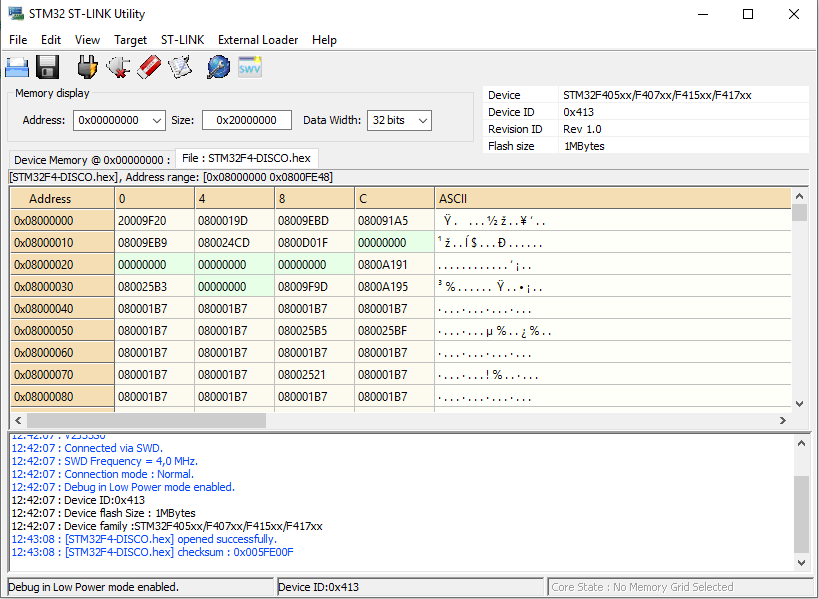
*Figure 31 Kit STM32F4*

## Khối Debug

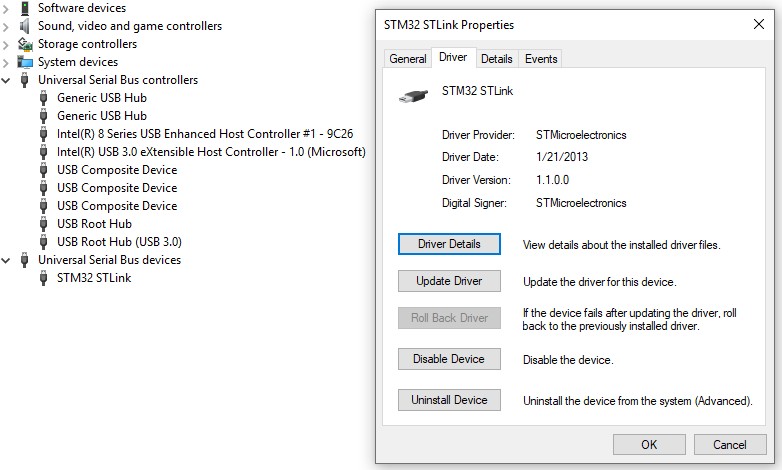
Để debug, kit STM32F4 hỗ trợ giao diện ST-LINK / V2 (hoặc V2-A) tích hợp trên mạch. Để gỡ lỗi, người dùng chỉ cần kết nối bộ phát triển với máy tính thông qua đầu nối USB. Giao diện này tương thích với IDE gần như phổ biến như KeilC uVison, IAR, Eclipse |

Để sử dụng lần đầu tiên, người dùng phải cập nhật trình điều khiển để kết nối thiết bị. Điều này rất dễ dàng do một công cụ được ST hỗ trợ, được gọi là STM32 ST Link Utility.

Mặt khác, ST Link cũng là một công cụ GUI mạnh mẽ để kết nối mạch, quan sát bộ nhớ, xóa bộ nhớ flash của chip…



*Figure 32 Giao diện STM32 ST Link*



*Figure 33 Cập nhập driver cho mạch*

## Khối xử lý âm thanh

Bộ vi điều khiển STM32F407VG sử dụng bộ xử lý âm thanh (CS43L22) để phát ra âm thanh thông qua đầu nối jack cắm âm thanh 3.5.

Bộ vi điều khiển STM32F407VG điều khiển bộ xử lý âm thanh thông qua giao diện I2C và xử lý tín hiệu số thông qua kết nối I2S hoặc tín hiệu đầu vào tương tự.

• Âm thanh có thể đến độc lập với các đầu vào khác nhau:

- Micrô ST-MEMS (MP45DT02) kỹ thuật số sử dụng giao thức PDM hoặc analog khi sử dụng bộ lọc thông thấp

* Bộ nhớ trong của vi điều khiển STM32F407VG
* Đầu nối USB: từ bộ nhớ ngoài lớn như khóa USB, USB HDD
  + Âm thanh có thể được phát ra theo nhiều cách khác nhau thông qua bộ xử lý âm thanh:

- Sử dụng giao thức I2S

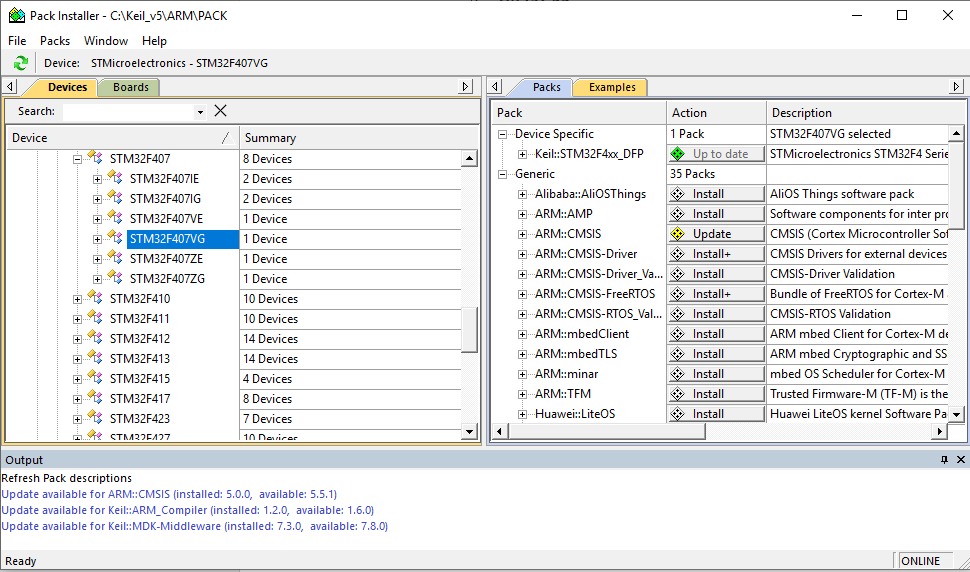
- Sử dụng DAC để đầu vào tương tự AIN1x của CS43L22

- Sử dụng đầu ra micrô trực tiếp qua bộ lọc thông thấp đến đầu vào tương tự AIN4x của CS43L22

## Keil C uVison 5 IDE

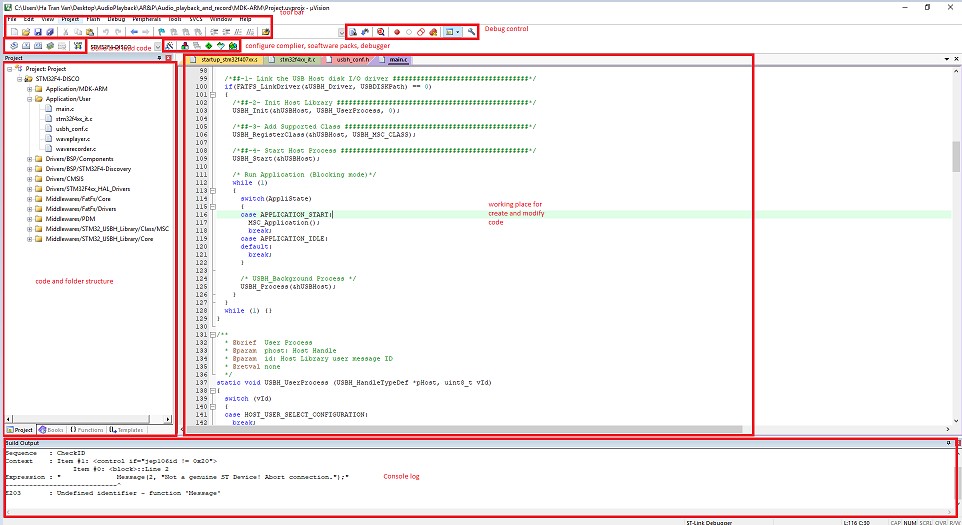
Keil C uVison 5 là một IDE mạnh mẽ để lập trình nhúng và gỡ lỗi. Nó hỗ trợ hầu hết các họ MCU của hầu hết các nhà sản xuất MCU như ST, NXP, TI, Microchip, Renesas.

Sau khi tải xuống và cài đặt, bây giờ chúng ta phải tải xuống gói phần mềm cho vi điều khiển mà chúng ta muốn sử dụng. Các gói này chứa phần mềm cơ bản mà chúng ta cần để khởi tạo MCU như startup code, để biên dịch và xây dựng chương trình như linker file hoặc thư viện thiết bị ngoại vi giúp chúng ta dễ dàng định cấu hình và sử dụng các thiết bị ngoại vi này. Công việc này có thể được thực hiện với trình cài đặt gói được cài đặt cùng với IDE. Tất cả chúng ta cần là chọn gói chúng ta muốn và nhấp vào cài đặt.



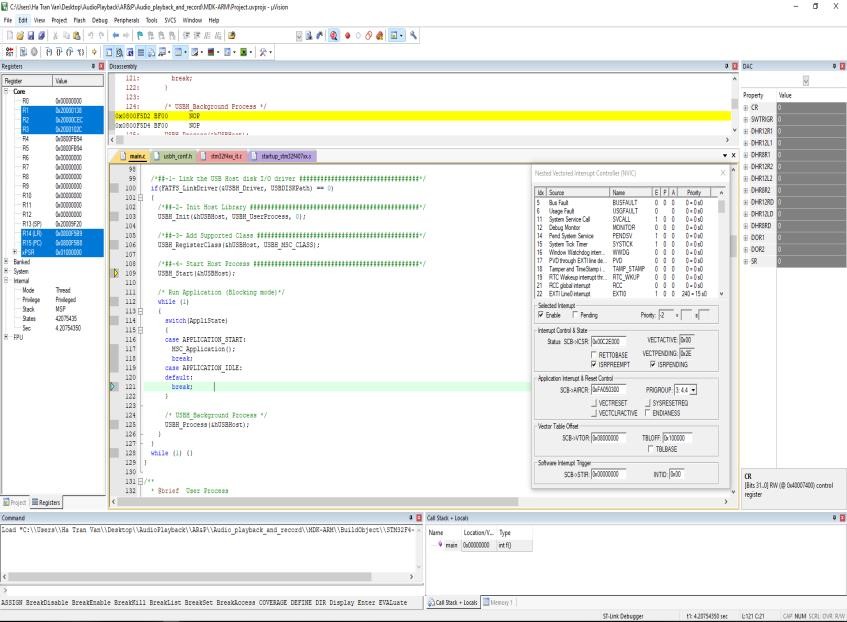
*Figure 34 Bộ cài đặt phần mềm Keil C*

Sau khi cài đặt phần mềm cần thiết, bây giờ chúng ta có thể tạo một project cho mục đích riêng của mình. Dưới đây là cửa sổ làm việc chính của Keil C uVison 5.



*Figure 35 Giao diện Keil C*

Keil C cũng hỗ trợ giao diện gỡ lỗi đẹp và mạnh. Nó cho phép nhà phát triển quan sát hành vi của mọi thiết bị ngoại vi, mã asm, trạng thái của thanh ghi lõi. Với các chức năng này, nhà phát triển có thể dễ dàng gỡ lỗi và làm cho code của họ hoạt động.

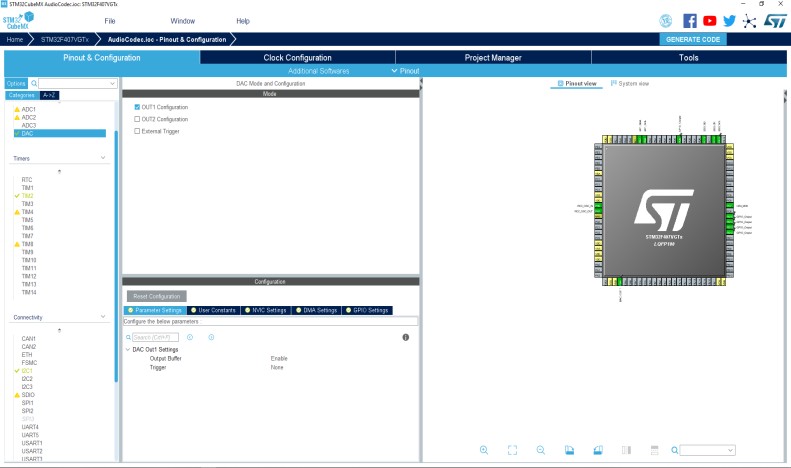


*Figure 36 Giao diện Keil C*

## STM32 CubeMx

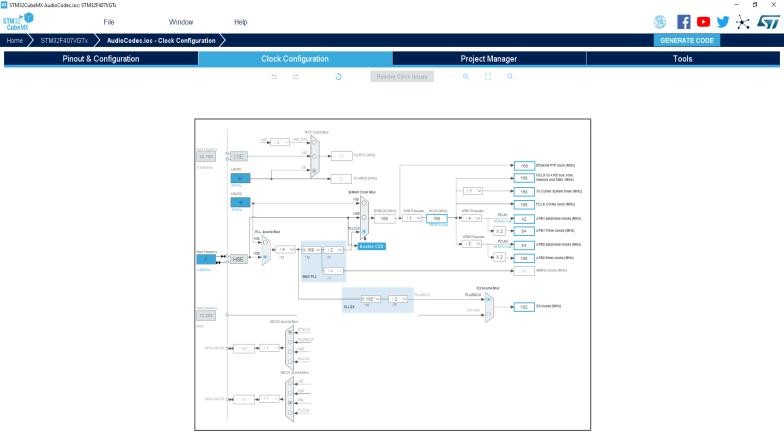
STM32 Cube Mx do ST phát triển để hỗ trợ người dùng cập nhật gói phần mềm cho sản phẩm của họ hoặc định cấu hình các thiết bị ngoại vi dễ dàng hơn. Phần mềm hữu ích này có thể được tải xuống từ trang web ST theo liên kết dưới đây:

<https://www.st.com/en/development-tools/stm32cubemx.html>

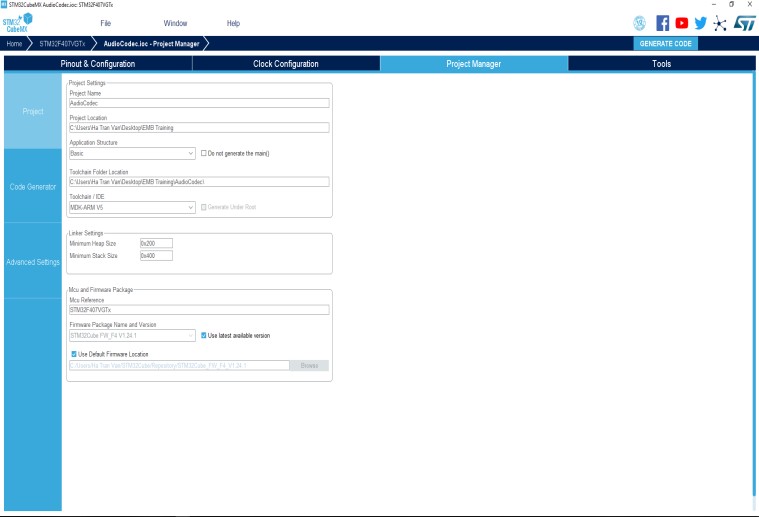


*Figure 37 ST Cube Mx GUI*

Sau khi chọn MCU, nhà phát triển có thể định cấu hình tất cả các đặc điểm anh ta muốn như chân GPIO, clock, linker, IDE…



*Figure 38 Cấu hình clock STM32 Cube Mx*



*Figure 39 Quản lý source code ST Cube Mx*

# CHAPTER IV: SYSTEM DESIGN

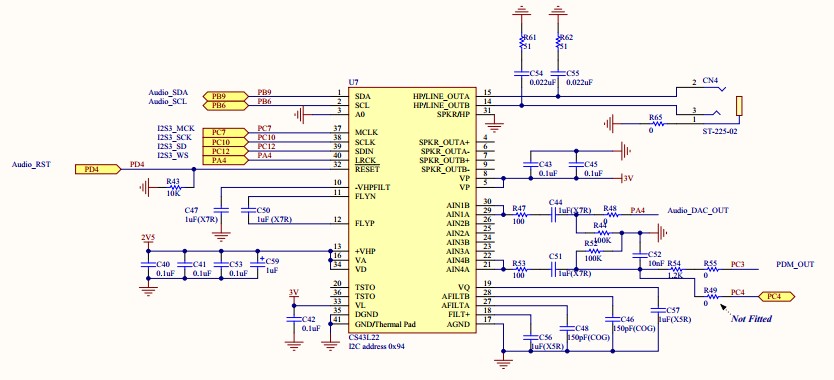
Trong chương này, tôi sẽ phân tích các chức năng của hệ thống, nhiệm vụ của mọi thành phần, thiết bị ngoại vi trong hệ thống và cách liên kết tất cả những thứ này để tạo ra một ứng dụng hoàn chỉnh.

Có ba phần chính trong chương này

* + Khái quát ứng dụng
  + Chức năng chơi nhạc
  + Chức năng ghi âm

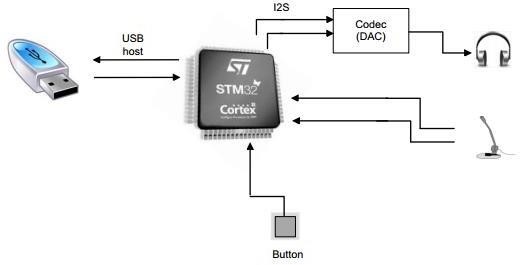
## Khái quát về ứng dụng

File nhac WAV lưu trữ trong USB có thể được chơi bởi kit STM32F4. Ứng dụng này sử dụng micro MEMS, bộ chuyển đối tương tự số DAC, tai nghe và USB.



*Figure 40 Sơ đồ nguyên lý khối xử lý âm thanh trên kit STM32F4 discovery*

USB phải được cấu hình ở chế độ host. Lớp lưu trữ lớn (MSC) được sử dụng để gửi / nhận dữ liệu âm thanh đến / từ USB. I2S được cấu hình ở chế độ master và được sử dụng để truyền dữ liệu âm thanh đến bộ chuyển đổi tín hiệu số sang tương tự (DAC) trước khi dẫn tín hiệu đến loa. DMA được sử dụng để gửi từ bộ đệm đến các thiết bị ngoại vi I2S giúp giảm tải CPU một cách hiệu quả. I2C được sử dụng để điều khiển các thiết bị bên ngoài như mã hóa, giải mã âm thanh và lấy dữ liệu từ thiết bị đó. Nút nhấn dành cho người dùng là chuyển đổi giữa hai chức năng phát lại hoặc ghi âm.



*Figure 41 Sơ đồ kết nối các thành phần*

## Ứng dụng chơi nhạc

Quy trình cho ứng dụng phát âm thanh trên kit

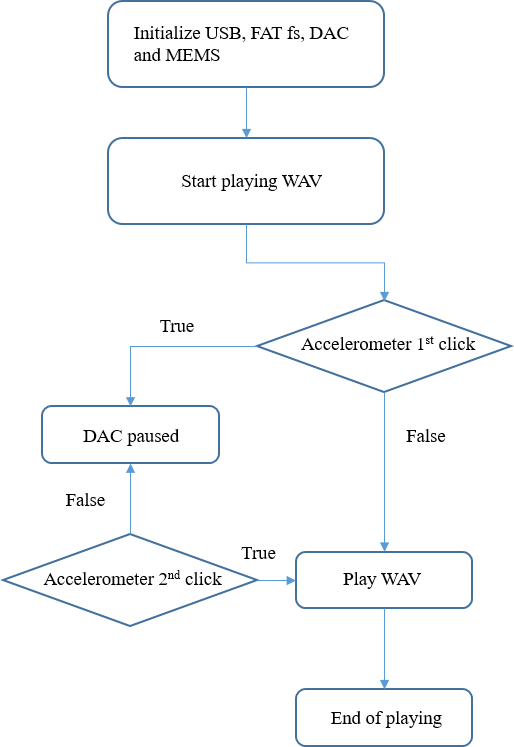
1. Kích hoạt USB, Hệ thống tệp Fat Fs, Audio DAC, MEMS microphone

2. Chuyển tệp WAV từ USB sang SRAM bên trong của MCU, nhận từng khối (1024 byte) bằng DMA trong bộ đệm đầu tiên.

3. DMA gửi dữ liệu trong bộ nhớ tới I2S để chuyển nó sang bộ chuyển đổi số tương tự (DAC).

4.Trong khi I2S, DAC xử lý dữ liệu trong bộ đệm đầu tiên, Dữ liệu được ghi vào bộ đệm thứ 2

5. Hai bộ đệm được hoán đổi vô thời hạn cho đến hết tập tin âm thanh



*Figure 42 Sơ đồ ứng dụng chơi nhạc*

## Ứng dụng ghi âm

Quy trình ghi âm:

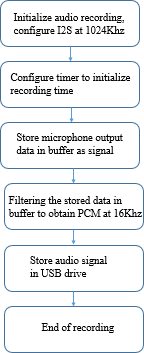
1. Khởi tạo bản ghi âm thanh: cấu hình I2S là 1024 Khz tạo xung đồng hồ cho micrô MEMS

2. Cấu hình hẹn giờ bằng timer để khởi tạo thời gian ghi

3. Lưu trữ tín hiệu thu từ micro trong bộ đệm

4. Lọc dữ liệu được lưu trữ để thu được tín hiệu PCM ở tần số lấy mẫu 16 KHz

5. Lưu trữ tín hiệu đã lọc vào USB.

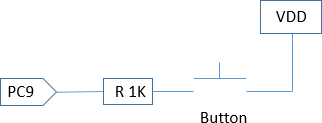


*Figure 43 Quy trình ghi âm*

Modun I2S đã được cấu hình ở chế độ chính để tạo chu kỳ xung nhịp chính xác (1.024 MHz). Đồng hồ 1.024 MHz có thể được tính từ luồng âm thanh đầu ra (16 Khz) và hệ số suy giảm (64) được chọn cho ứng dụng này (16000 Hz x 64 = 1.024 MHz).

## Điều khiển âm thanh

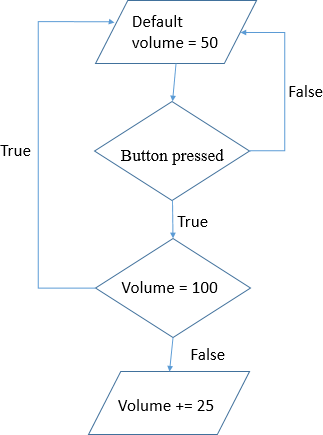
Sơ đồ kết nối của nút nhấn điều khiển âm thanh



*Figure 44 Sơ đồ kết nối nút nhấn điều khiển âm thanh*

Để phát hiện trạng thái nút, tôi thiết lập chân MCU (PC9) ở chế độ detect ngắt ngoài (sườn lên ). Theo mặc định, chân MCU ở chế độ kéo xuống nên mỗi khi nhấn nút, VDD được kết nối với PC9. Điều này làm cho nó được kéo lên cao và cũng tạo ra một sườn lên trên pin này. Sự kiện này gây ra ngắt và trình xử lý ngắt sẽ được gọi để phục vụ sự kiện này.

Sự thay đổi âm lượng do trạng thái nút được mô tả trong sơ đồ bên dưới..



*Figure 45 Sơ đồ điều khiển âm thanh*

# CHAPTER V: Kết Luận và Tài Liệu Tham Khảo

**Kết Luận**

Khi làm việc với đồ án này, tôi đã có cơ hội thực hành với vi điều khiển (cách thức hoạt động của ngắt, cách chuyển đổi xung đồng hồ cho từng mô-đun ngoại vi, cách thức thực hiện một lệnh và v.v. ứng dụng. Không chỉ có kiến ​​thức về cấu trúc vi điều khiển, tôi đã có cơ hội cải thiện kỹ năng lập trình C của mình khi gỡ lỗi. Tất cả mang lại cho tôi nhiều kinh nghiệm quý giá trong việc tạo ra một sản phẩm nhúng. Trong tương lai, tôi sẽ tiếp tục làm việc với đồ án này để phát triển nhiều chức năng hơn, ví dụ như, kết nối với chipset Bluetooth để truyền dữ liệu đến loa hoặc máy tính xách tay để phát trực tiếp.

Mặc dù tôi đã cố gắng hết sức nhưng do những hạn chế về cả kiến ​​thức vi điều khiển và kỹ năng mã hóa, tôi không thể tránh được những sai lầm. Mong nhận dduwwocj sự góp ý của quý thầy cô.

Một lần nữa, tôi muốn bày tỏ lòng biết ơn đến người thầy của mình, Tiến sĩ Dung Le. Thầy giúp tôi tìm chủ đề của luận án và tạo ra các phương pháp để tôi giải quyết các vấn đề và đi đến cuối cùng. Nếu không có sự giúp đỡ của thầy, tôi sẽ hoàn thành đồ án này.

Ha Noi, 06/10/ 2019

Tran Van Ha

# Tài liệu tham khảo

https://[www.st.com/content/ccc/resource/technical/document/user\_manual/70/fe/4a/3f/e7/e1/4f/7d/DM000](http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM000) 39084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf

Truy cập lần cuối May/20/2019.

https://[www.st.com/content/ccc/resource/technical/document/reference\_manual/3d/6d/5a/66/b4/99/40/d4/](http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/) DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf truy cập lần cuối May/27/2019

[https://www.nxp.com/docs/en/user-guide/UM10204.pdf,](https://www.nxp.com/docs/en/user-guide/UM10204.pdf)  truy cập lần cuối May/25/2019.

<https://en.wikipedia.org/wiki/Direct_memory_access>, truy cập lần cuối May/15/2019

<https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>, truy cập lần cuối May/18/2019.

<https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system>, truy cập lần cuối May/15/2019.

<https://en.wikipedia.org/wiki/WAV>, truy cập lần cuối May/10/2019.

[https://www.st.com/en/evaluation-tools/stm32f4discovery.html#resource,](https://www.st.com/en/evaluation-tools/stm32f4discovery.html#resource)  truy cập lần cuối May/29/2019.

<https://github.com/fboris>. truy cập lần cuối May/29/2019.