



**Triển khai bộ thu cho các giao thức điều khiển từ xa hồng ngoại
sử dụng vi điều khiển STM32F10xxx**

Giới thiệu

Bức xạ hồng ngoại là vùng của phổ điện từ nằm giữa vi sóng và ánh sáng nhìn thấy.

Bức xạ hồng ngoại có hai dãy. Ánh sáng hồng ngoại gần có bước sóng gần nhất với ánh sáng nhìn thấy và tia hồng ngoại xa gần vùng vi sóng của phổ điện từ hơn.

Các sóng ngắn hơn là những sóng được sử dụng bởi điều khiển từ xa. Thông tin được truyền và nhận bằng năng lượng điện từ, không sử dụng dây dẫn.

Công nghệ hồng ngoại mang lại những lợi thế quan trọng như một hình thức truyền thông không dây. Ngày nay, hầu hết tất cả các thiết bị âm thanh và video đều có thể được điều khiển bằng điều khiển từ xa hồng ngoại. Ở đầu nhận, một máy thu phát hiện các xung ánh sáng, các xung này được xử lý để lấy / giải mã thông tin mà chúng chứa.

Có nhiều giao thức hồng ngoại phổ biến được sử dụng để truyền dữ liệu qua đèn hồng ngoại, chẳng hạn như RC5, SIRC, ...

Mục đích của ghi chú ứng dụng này là cung cấp giải pháp chung để triển khai bộ thu IR trong phần mềm sử dụng vi điều khiển STM32F10xxx. Ví dụ về triển khai phần mềm được cung cấp cho các giao thức RC5 và SIRC, các giao thức khác được hỗ trợ và có sẵn theo yêu cầu (để biết thêm thông tin, hãy liên hệ với văn phòng bán hàng STMicroelectronics tại địa phương của bạn).

Nội dung

1	Cân nhắc phần cứng.	3
2	Giải pháp chung.	4
3	Giải pháp giao thức RC5.	5
3.1	Thông tin cơ bản về giao thức.	5
3.2	Triển khai phần mềm bằng một GPIO duy nhất với một bộ đếm thời gian chung.	7
3.2.1	Cơ chế đọc khung RC5.	7
3.3	Triển khai phần mềm bằng cách sử dụng GP-Timer được định cấu hình ở chế độ đầu vào PWM.	10
3.3.1	Cơ chế giải mã RC5 Frame. Thư viện 10 RC5.	
3.3.2 13 Cách sử dụng trình điều khiển bộ giải mã RC5.	
3.3.3 14	
3.4	So sánh các giải pháp RC5.	15
4	Giải pháp điều khiển hồng ngoại SIRC.	17
4.1	Thông tin cơ bản về giao thức.	17
4.2	phần mềm.	19
4.3 21 Cách sử dụng trình điều khiển bộ giải mã SIRC.	
4.4 22	
5	Lớp giao diện.	24
6	Cách sử dụng trình điều khiển IR.	26
6.1	Các chương trình trình diễn.	26
6.1.1	Demo sử dụng GP-Timer được cấu hình ở chế độ PWM.	26
6.1.2	GPIO duy nhất với bộ đếm thời gian chung.	26
7	Cách tùy chỉnh các trình điều khiển IR.	28
số 8	Phản kết luận.	31
9	Lịch sử sửa đổi.	32

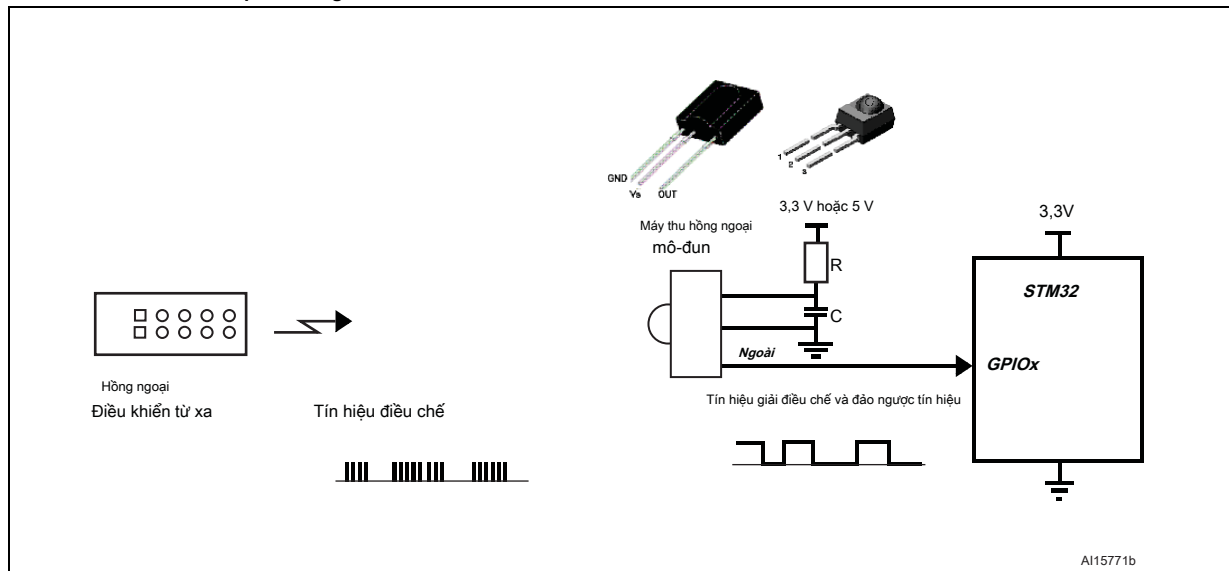
1 Cân nhắc phần cứng

Để cải thiện khả năng loại bỏ tiếng ồn, các xung IR được điều chế ở khoảng 36 kHz, 38 kHz hoặc 40 kHz. Cách dễ nhất để nhận các xung này là sử dụng IR-mô-đun bộ thu / giải điều chế như TSOP1736 (phiên bản nguồn cung cấp 5 V) hoặc TSOP34836 (phiên bản nguồn cung cấp 3,3 V) hoặc số bộ phận tương đương khác (tham khảo [Hình 1](#)).

Đây là các thiết bị 3 chân nhận chùm tia hồng ngoại và xuất ra dòng bit đã được giải điều chế trên chân đầu ra được kết nối trực tiếp với một trong các chân GPIO của vi điều khiển STM32 hoặc các kênh GP-Timers Input Capture. Nếu TSOP1736 được sử dụng, GPIO được chọn phải là Dung lượng năm volt (FT). Đầu ra của mô-đun IR bị đảo ngược so với dữ liệu được truyền (dữ liệu ở trạng thái nhân rồi cao và logic '0' trở thành logic '1' và ngược lại).

Ghi chú: *Mô-đun IR cần hai thành phần bên ngoài: tụ điện và điện trở (tham khảo biểu dữ liệu mô-đun IR liên quan để biết các giá trị của chúng).*

Hình 1. Cấu hình phần cứng



2 Giải pháp chung

Mỗi giao thức Hồng ngoại có thể được giải mã bằng cách sử dụng một trong các thiết bị ngoại vi hẹn giờ được nhúng trong vi điều khiển STM32. Bộ định thời này có thể được cấu hình ở chế độ đầu vào PWM và được sử dụng để lấy mẫu các bit khung hồng ngoại. Chức năng chụp đầu vào bộ hẹn giờ đang hoạt động trên các cạnh có cực tính đối diện.

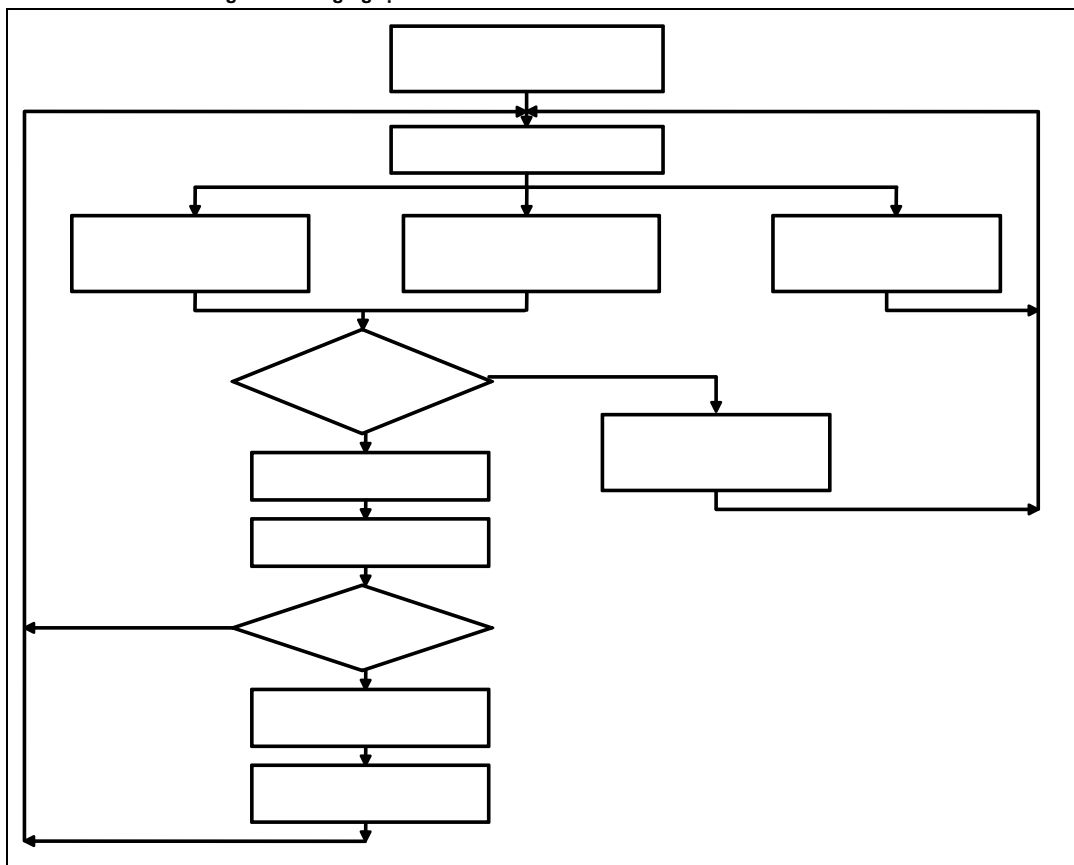
Bộ định thời tạo ra ba loại ngắt:

- Ngắt tại mỗi cạnh rơi: Điều này có thể được sử dụng để đo toàn bộ xung (khoảng thời gian giữa hai cạnh rơi liên tiếp).
- Ngắt tại mỗi cạnh tăng: Điều này có thể được sử dụng để đo xung thấp (khoảng thời gian giữa các cạnh giảm và cạnh tăng).
- Sự kiện cập nhật: Được sử dụng để đưa gói Hồng ngoại vào trạng thái mặc định (Số bit, Dữ liệu và Trạng thái) khi bộ đếm Timer bị tràn.

Xung thấp và toàn bộ thời gian xung được sử dụng để xác định giá trị bit. Nếu khoảng thời gian nằm trong phạm vi dung sai của thời gian bit, chúng tôi xác định giá trị bit (Logic0, Logic1 hoặc Header).

Lưu đồ dưới đây cung cấp một cái nhìn tổng quan về quy trình giải mã Hồng ngoại.

Hình 2. Lưu đồ giải mã hồng ngoại

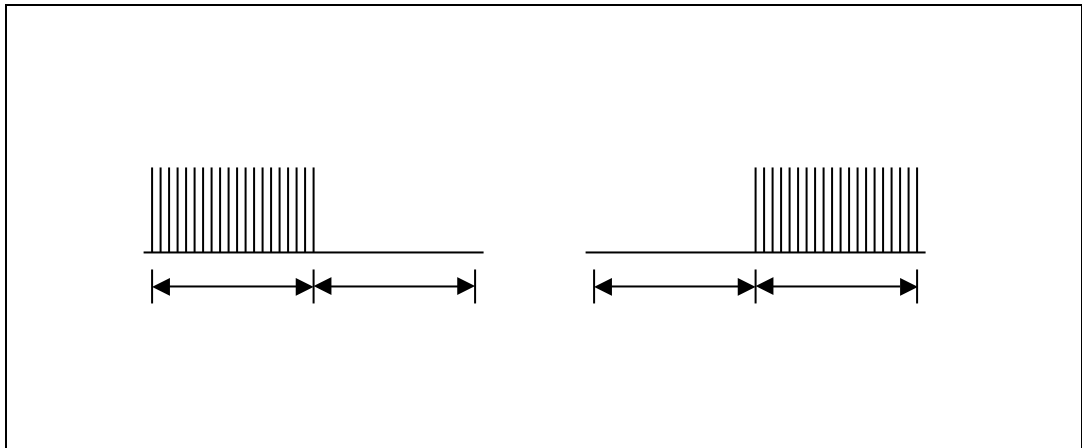


3 Giải pháp giao thức RC5

3.1 Khái niệm cơ bản về giao thức

Mã RC5 là một từ 14 bit, nó sử dụng điều chế hai pha (còn gọi là mã hóa Manchester) tần số sóng mang IR 36 kHz. Tất cả các bit đều có độ dài bằng nhau là 1,778 ms, với một nửa thời gian bit được lấp đầy bởi sự bùng nổ của sóng mang 36 kHz và nửa còn lại là không hoạt động. Một số 0 logic được biểu thị bằng một cụm trong nửa đầu của thời gian bit. Một logic được biểu diễn bằng một cụm trong nửa sau của thời gian bit. Chu kỳ hoạt động của tần số sóng mang 36 kHz là 33% hoặc 25%, giúp giảm tiêu thụ điện năng.

Hình 3. Biểu diễn bit RC5

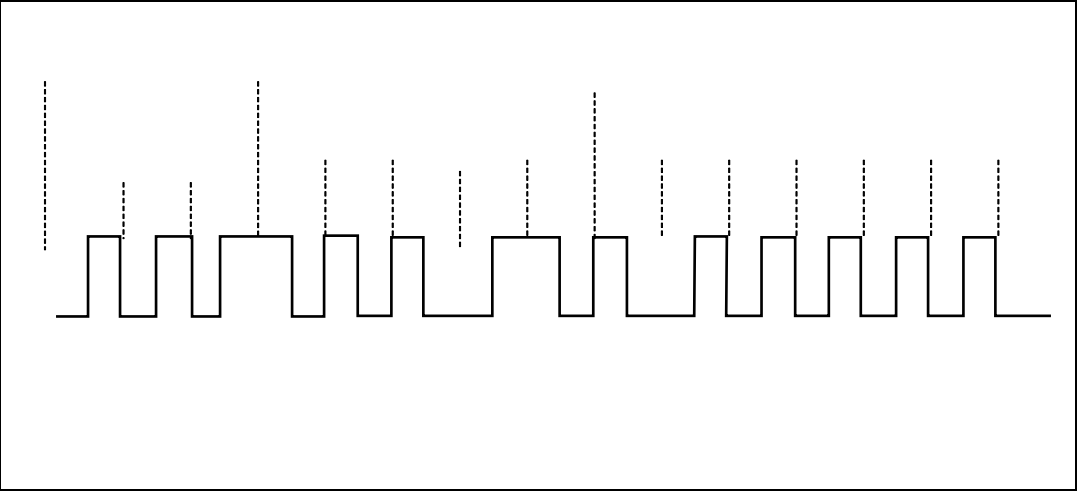


Khung RC5 có thể tạo ra 2048 (32x64) lệnh khác nhau được tổ chức trên 32 nhóm. Mỗi nhóm có 64 lệnh khác nhau.

Khung RC5 chứa các trường sau. Một ví dụ về khung RC5 được hiển thị trong [hình 4](#).

- **Bắt đầu bit (S):** Độ dài 1 bit, luôn là logic 1.
- **Trường bit (F):** Độ dài 1 bit, cho biết lệnh được gửi ở trường dưới (logic 1 = 0 đến 63 thập phân) hay trường trên (logic 0 = 64 đến 127 thập phân). Bit trường được thêm vào sau đó khi nhận ra rằng không đủ 64 lệnh trên mỗi thiết bị. Trước đây, bit trường được kết hợp với bit bắt đầu. Nhiều thiết bị vẫn sử dụng hệ thống gốc này.
- **Bit điều khiển hoặc bit chuyển đổi (C):** Độ dài 1 bit, sẽ chuyển đổi mỗi khi nhấn nút. Điều này cho phép thiết bị nhận phân biệt giữa hai lần nhấn nút liên tiếp (chẳng hạn như "1", "1" cho "11").
- **Địa chỉ:** Độ dài 5 bit, chọn một trong 32 hệ thống có thể.
- **Chỉ huy:** Độ dài 6 bit, (kết hợp với bit trường) đại diện cho một trong 128 lệnh RC5 có thể.

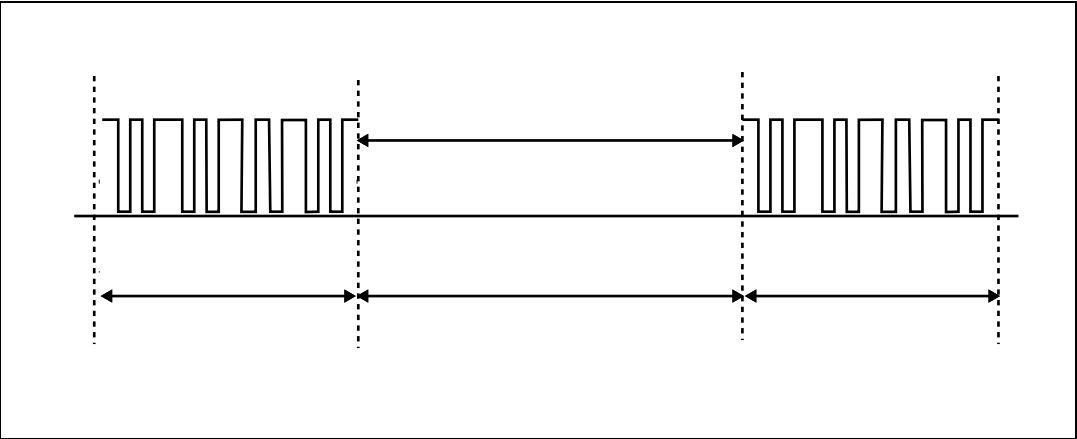
Hình 4. Ví dụ về khung RC5



Để tránh xung đột khung, thời gian nhàn rỗi được chèn vào giữa hai khung liên tiếp có chiều rộng cụ thể (xem [Hình 5](#)).

Thời gian nhàn rỗi được định nghĩa là rộng 50 bit. Vì vậy, chu kỳ của khung có chiều rộng 64 x 1 bit:
 $64 \times 1,778 = 113,792 \text{ ms}$.

Hình 5. RC5 thời gian nhàn rỗi



Bảng 1. Thời gian RC5

Sự miêu tả	Tối thiểu.	Điện hình	Tối đa
RC5 Nửa chu kỳ bit RC5 Chu	640 μs	889 μs	1140 μs
kỳ toàn bộ bit Thời gian thông	1340 μs	1778 μs	2220 μs
báo RC5	23,644 mili giây	24,889 μs	26.133 mili giây
Thông điệp RC5 thời gian lặp lại	108,089 mili giây	113,778 mili giây	119,467 mili giây
Thời gian bit xung sóng mang	27,233 μs	27,778 μs	28.349 μs

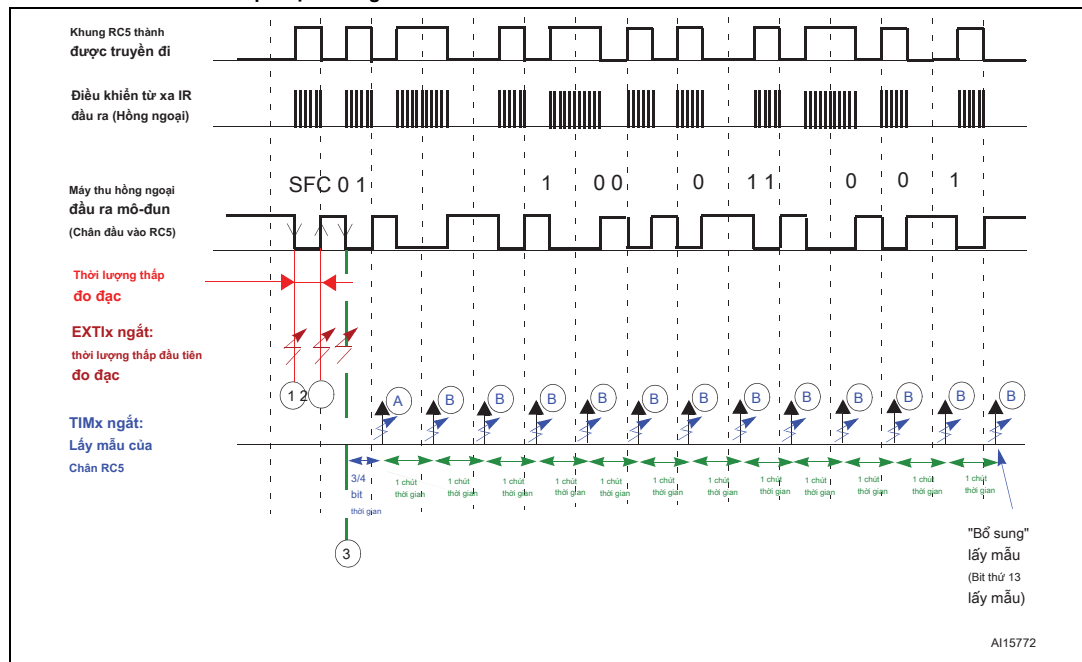
3.2 Triển khai phần mềm bằng cách sử dụng một GPIO duy nhất với bộ đếm thời gian chung

3.2.1 Cơ chế đọc khung RC5

Hình 6 hiển thị cách nhận khung RC5. Về cơ bản, hai trong số các thiết bị ngoại vi nhúng của vi điều khiển STM32 được sử dụng cho mục đích này: EXTI và bộ đếm thời gian (TIMx).

Chân STM32 được kết nối với chân đầu ra của mô-đun IR có thể là bất kỳ GPIO nào do người dùng chọn.

Hình 6. Cơ chế tiếp nhận khung RC5



Sự kiện ngắt EXTI

Ngắt EXTI được sử dụng để bắt đầu và dừng TIMx nhằm đo khoảng thời gian thấp đầu tiên để xác thực thời gian tiêu đề của khung RC5:

- Sự kiện ngắt EXTI đầu tiên (1): bộ đếm TIMx được khởi tạo và kích hoạt.
- Sự kiện ngắt EXTI thứ hai (2): bộ đếm TIMx bị vô hiệu hóa, đọc và sau đó khởi tạo.

Giá trị được đọc từ bộ đếm cho biết khoảng thời gian đo được. Lần thực hiện thứ 3 của ngắt EXTI phụ thuộc vào khoảng thời gian đo được:

- Nếu thời lượng nằm trong phạm vi dung sai của một nửa thời gian bit, EXTI không bị vô hiệu hóa và ngắt EXTI xảy ra lần thứ 3, điều này cho phép TIMx. Sau đó TIMx bắt đầu lấy mẫu dữ liệu RC5. Trong trường hợp này, bit Trường được nhận dạng là logic 1.

- Nếu thời lượng nằm trong phạm vi dung sai của thời gian một bit, thì EXTI sẽ bị vô hiệu hóa tại thời điểm này và ngắt sự kiện cập nhật TIMx cũng như bộ đếm TIMx được bật để bắt đầu lấy mẫu dữ liệu RC5. Trong trường hợp này, bit Trường được nhận dạng là logic 0. Nếu thời lượng dường như có trục trặc, hệ thống sẽ được khởi tạo cho khung RC5 tiếp theo.
- Sự kiện ngắt EXTI thứ ba (3): sự xuất hiện ngắt này phụ thuộc vào khoảng thời gian của khoảng thời gian thấp đầu tiên, xem (2). Khi xảy ra ngắt, TIMx được kích hoạt và bắt đầu lấy mẫu dữ liệu RC5.

Sự kiện gián đoạn TIMx

TIMx được sử dụng để lấy mẫu từng bit của khung RC5 sau khi kiểm tra thời gian của khoảng thời gian thấp đầu tiên của khung.

Ngắt TIMx được thực thi 13 lần trong một khung RC5 để lấy mẫu tất cả các bit.

Bit Start (S) và bit Field (F) không được TIMx lấy mẫu và một "bit bổ sung" được lấy mẫu ở cuối khung RC5 để đảm bảo rằng tất cả các bit đã được nhận và trạng thái nhân rồi hiện có.

- Sự kiện ngắt TIMx (A): tại thời điểm này, chân RC5 được lấy mẫu bằng một lần đọc thanh ghi dữ liệu đầu vào GPIO. Trong quy trình dịch vụ ngắt này, TIMx được cấu hình để tạo ngắt định kỳ mỗi thời gian bit.
- Sự kiện ngắt TIMx (B): tại thời điểm này, chân RC5 được lấy mẫu bằng một lần đọc thanh ghi dữ liệu đầu vào GPIO và quy trình dịch vụ ngắt sẽ kiểm tra xem số lượng bit dữ liệu đã đạt đến 13 chưa ($n = 13: 14 - 2 + 1$). Nếu có, bộ đếm TIMx và ngắt cập nhật TIMx bị vô hiệu hóa.

Như chúng ta thấy, việc đọc từ thanh ghi dữ liệu đầu vào GPIO phản ánh trực tiếp giá trị của bit. Nếu giá trị đọc ở mức thấp, điều này ngụ ý rằng giá trị bit là logic '0'. Nếu giá trị được đọc ở mức cao, điều này ngụ ý rằng giá trị bit là logic '1'.

Cách sử dụng thư viện RC5

Trình điều khiển RC5 rất đơn giản để sử dụng. Có bốn chức năng có sẵn cho người dùng.

RC5_Receiver_Init ()

Chức năng này nhằm khởi tạo các thiết bị ngoại vi khác nhau được trình điều khiển RC5 sử dụng: GPIO, EXTI và TIMx. Nó phải được gọi sau cấu hình đồng hồ người dùng.

RC5_Sample_Data ()

Hàm này được sử dụng để lấy mẫu dữ liệu RC5. Nó phải được gọi trong quy trình RC5_TIM_IRQ_Handler (TIMx_IRQHandler) trong tệp stm32f10x_it.c. Tệp RC5_IR_Receiver.h phải được bao gồm trong tệp stm32f10x_it.c. Theo mặc định, TIM2 được sử dụng. Bạn có thể sử dụng bất kỳ bộ hẹn giờ nào bằng cách sửa đổi các định nghĩa trong tệp RC5_IR_Emul_Receiver.h (đường dẫn: \Project \ InfraRed \ RC5_Decoding_TIM_EXTI \ inc) như sau:

Thí dụ:

Nếu bạn muốn sử dụng TIM3, hãy thực hiện các sửa đổi này (được đánh dấu trong **Dùng cam**):

```
# xác định RC5_TIM TIM 3
# xác định RC5_TIM_CLK RCC_APB1Periph_TIM 3
# xác định RC5_TIM_IRQn TIM 3_IRQn
# xác định RC5_TIM_IRQ_Handler TIM 3_IRQHandler
Bạn có thể chọn bất kỳ bộ hẹn giờ gia đình STM32F10x nào.
```


RC5_MeasureFirstLowDuration ()

Chức năng này đo lường và xác nhận thời lượng thấp đầu tiên của khung RC5. Khi thời gian này nằm trong phạm vi thời gian cho phép, chức năng sẽ bật lấy mẫu khung RC5. Hàm này phải được gọi trong trình xử lý ngắt EXTI thích hợp (trong tệp stm32f10x_it.c) tùy thuộc vào GPIO được sử dụng cho chân đầu vào RC5.

Theo mặc định, GPIOB.01 được sử dụng làm chân đầu vào RC5. Bạn có thể sử dụng bất kỳ bộ hẹn giờ nào bằng cách sửa đổi các định nghĩa trong tệp RC5_IR_Emul_Receiver.h

(đường dẫn: \ Project \ InfraRed \ RC5_Decoding_TIM_EXTI \ inc).

Thí dụ:

Nếu bạn muốn sử dụng GPIOD.09, hãy thực hiện các sửa đổi này (được đánh dấu trong **Dùng cảm**):

```
# định nghĩa    RC5_GPIO_PORT GPIO D

# định nghĩa    RC5_GPIO_CLK RCC_APB2Periph_GPIO D

# định nghĩa    RC5_GPIO_PIN GPIO_Pin_ 9

# định nghĩa    RC5_EXTI_PORT_SOURCE GPIO_PortSourceGPIO D

# định nghĩa    RC5_EXTI_PIN_SOURCE GPIO_PinSource 9

# định nghĩa    RC5_EXTI_IRQn EXTI_ 9_5_ IRQn

# định nghĩa    RC5_EXTI_LINE EXTI_Line 9

# định nghĩa    RC5_EXTI_IRQ_Handler EXTI_ 9_5_ IRQHandler
```

RC5_MeasureFirstLowDuration phải được gọi trong RC5_EXTI_IRQHandler.

RC5_Decode ()

Chức năng này nhằm mục đích được gọi trong ứng dụng người dùng. Nó giải mã các tin nhắn nhận được RC5. Nó trả về một cấu trúc chứa các giá trị khác nhau của khung RC5.

```
typedef struct
{
    __IO uint8_t ToggleBit; /* Chuyển đổi trường bit */ __IO uint8_t Địa chỉ; /* Trường địa
    chỉ */ __IO uint8_t Lệnh; /* Trường lệnh */ } RC5Frame_TypeDef;
```

RC5_decode () phải được gọi khi cờ IR_FrameReceive bằng YES. Thí dụ:

```
/* Cấu hình đồng hồ hệ thống */ RCC_Configuration ();

/* Khởi tạo nhận RC5 */ RC5_Receiver_Init ();

trong khi (1)
{
    /* Nếu khung RC5 đã được nhận, thì hãy giải mã nó */ if (IR_FrameReceive == YES)

    {
        /* Lấy khung RC5 */ RC5_Frame =
        RC5_Decode ();
    }
}
```

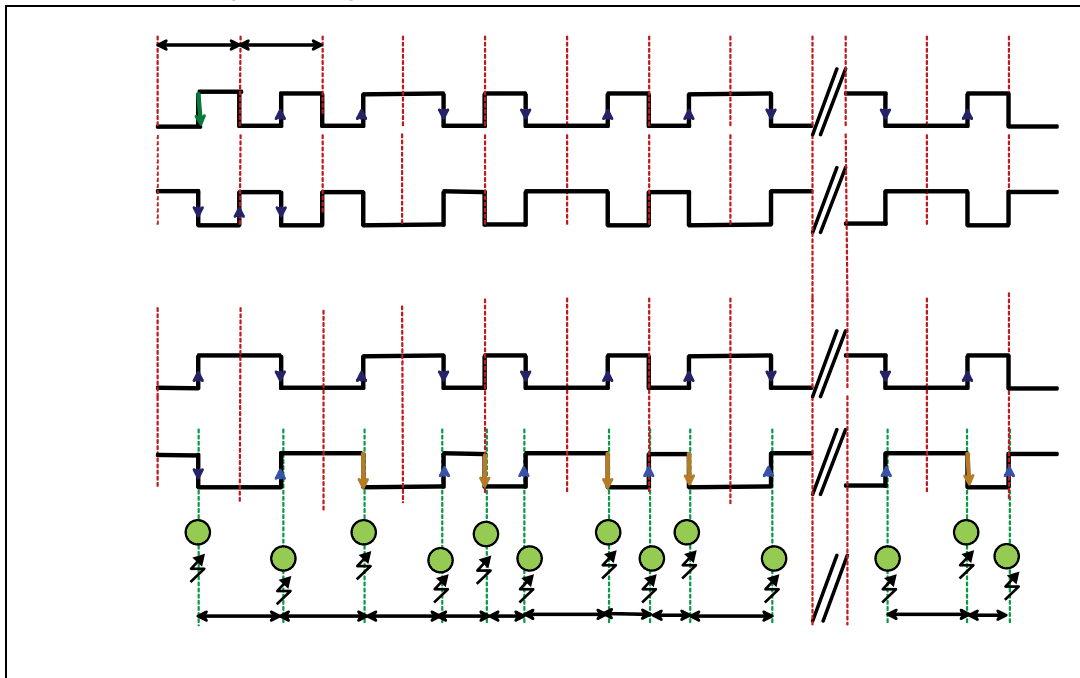
3,3 Triển khai phần mềm bằng cách sử dụng GP-Timer được định cấu hình ở chế độ đầu vào PWM

3.3.1 Cơ chế giải mã khung RC5

Hình 7 hiển thị cách nhận khung RC5. Một trong những thiết bị ngoại vi nhúng của vi điều khiển STM32 được sử dụng cho mục đích này: TIMER được cấu hình ở đầu vào PWM ở chế độ.

Đầu vào này có thể nắm bắt giá trị bộ đếm thời gian hiện tại ở cả cạnh giảm và cạnh lên cũng như tạo ra một ngắt trên cả hai cạnh. Tính năng này giúp bạn dễ dàng đo thời gian cao và thấp của xung RC5.

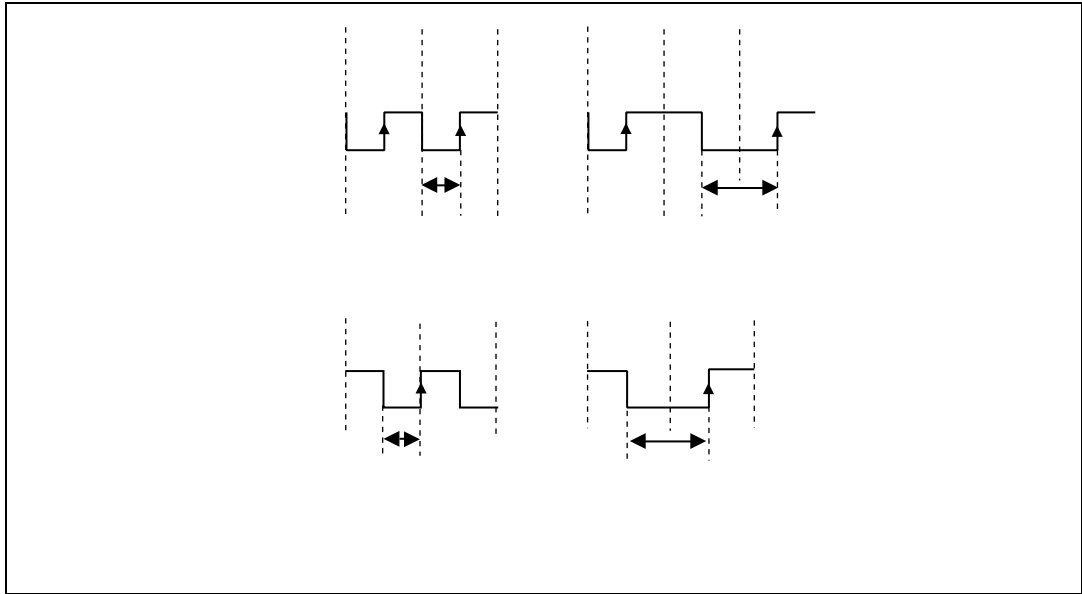
Hình 7. Cơ chế giải mã khung RC5



- Sự kiện ngắt TIMER: Rơi cạnh**
A: Ngắt TIMER được sử dụng để đo khoảng thời gian giữa hai cạnh rơi liên tiếp (Toàn bộ khoảng thời gian xung).
- Sự kiện ngắt TIMER: Cạnh tranh**
B: TIMER được sử dụng để đo khoảng thời gian giữa các cạnh giảm và tăng (Thời lượng xung thấp).

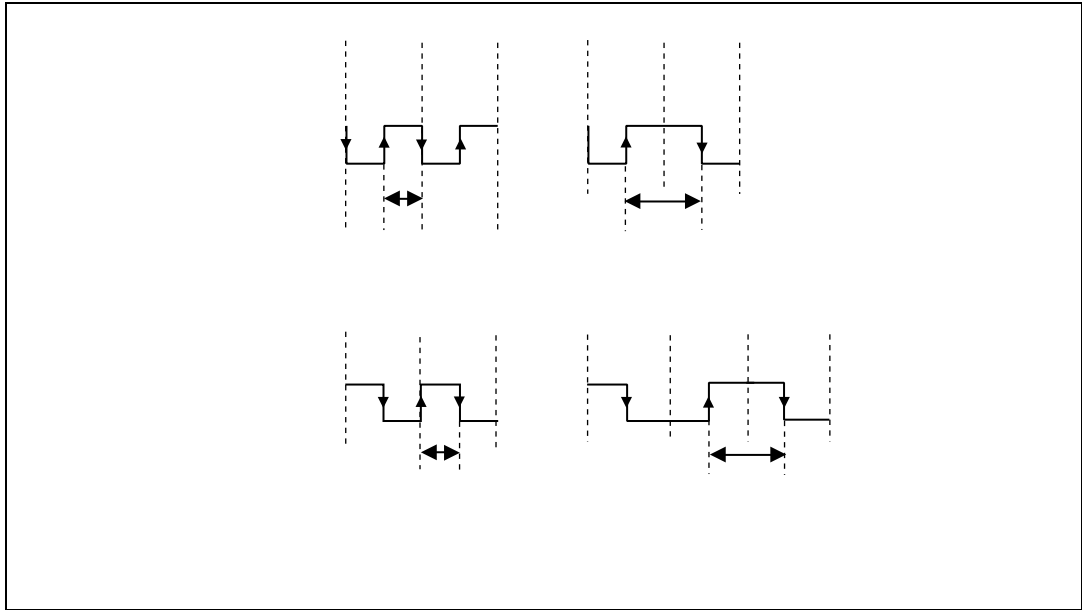
Hai khoảng thời gian được sử dụng để xác định giá trị bit. Mỗi giá trị bit được xác định liên quan đến bit cuối cùng.

Hình 8. Xác định bit bằng cạnh lên: xung thấp



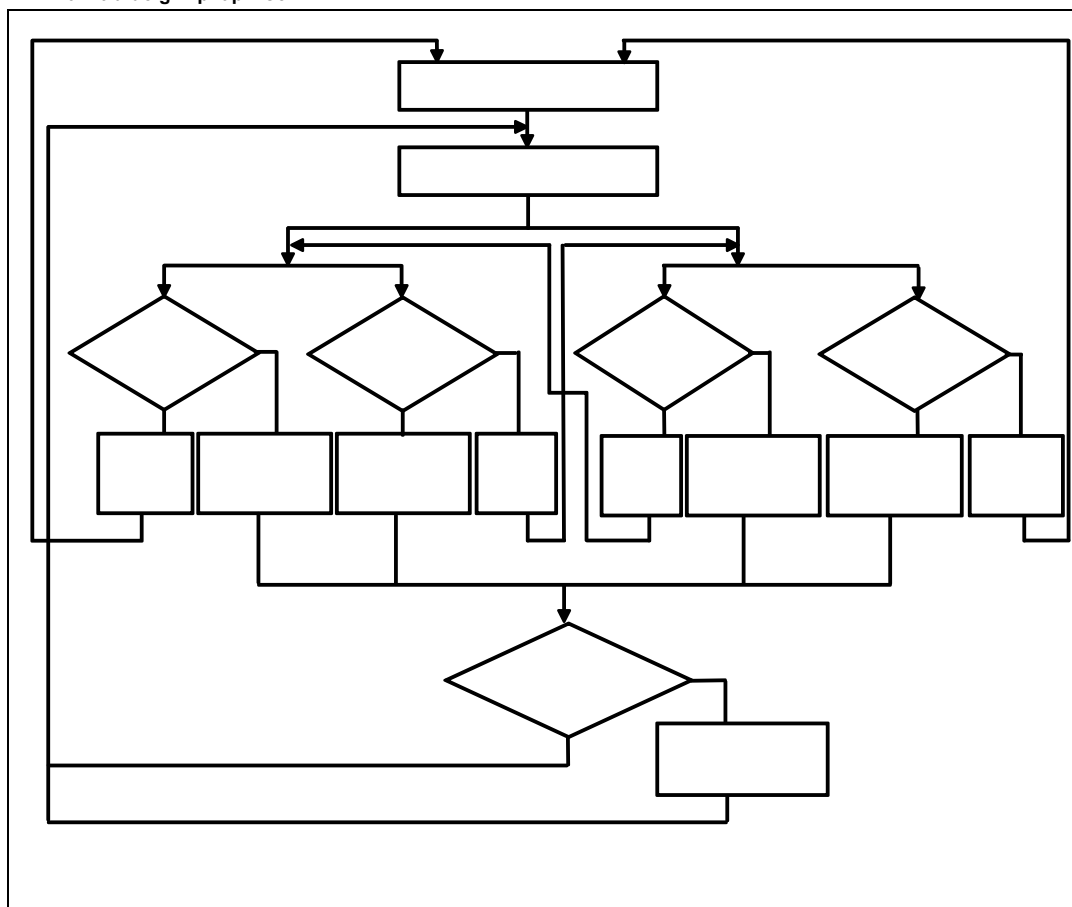
- Nếu thời lượng xung thấp bằng T và bit cuối cùng được xác định là '0' => bit thực là **'logic0'**.
- Nếu thời lượng xung thấp bằng $2T$ và bit cuối cùng được xác định là '0' => bit thực là **'Inv'**. (Trường hợp không hợp lệ: trường hợp này không thể được phát hành)
- Nếu thời lượng xung thấp bằng T và bit cuối cùng được xác định là '1' => bit thực là **'Nan'**. (Không bit: bit này được xác định ở cạnh rơi tiếp theo)
- Nếu thời lượng xung thấp bằng $2T$ và bit cuối cùng được xác định là '1' => bit thực là **'logic0'**.

Hình 9. Xác định bit bằng cạnh rơi: xung cao



- Nếu thời lượng xung cao bằng T và bit cuối cùng được xác định là '0' => bit thực là **'Nan'**. (Không có bit: bit này được xác định ở cạnh lên tiếp theo)
- Nếu thời lượng xung cao bằng 2T và bit cuối cùng được xác định là '0' => bit thực là **'logic1'**.
- Nếu thời lượng xung cao bằng T và bit cuối cùng được xác định là '1' => bit thực là **'logic1'**.
- Nếu thời lượng xung cao bằng 2T và bit cuối cùng được xác định là '1' => bit thực là **'Inv'**. (Trường hợp không hợp lệ: trường hợp này không thể được phát hành)

Hình 10. Lưu đồ giải pháp RC5



3.3.2

Thư viện RC5

Trình điều khiển RC5 rất đơn giản để sử dụng.

IR_RC5_Init ()

Chức năng này khởi tạo các thiết bị ngoại vi khác nhau (GPIO, TIMER, NVIC, ...).

IR_RC5_ResetPacket ()

Hàm này đặt cấu trúc gói ở trạng thái mặc định. Về cơ bản, hàm này được gọi trong quy trình TIM3_IRQHandler. Nó xảy ra mỗi lần tràn TIMER để đặt lại gói RC5.

IR_RC5_Decode (IRFrame_TypeDef * ir_frame)

Chức năng này nhằm mục đích được gọi trong ứng dụng người dùng. Nó giải mã các tín hiệu nhận được RC5. Cấu trúc sau đây chứa các giá trị khác nhau của khung RC5.

```
typedef struct
{
    __IO uint8_t FieldBit;           /* Trường bit trường / * Chuyển đổi * /
    __IO uint8_t ToggleBit;         trường bit / * Trường địa chỉ * /
    __IO uint8_t Địa chỉ;           * /
    __IO uint8_t Lệnh;              /* Trường lệnh * /
} IRFrame_TypeDef;
```

IR_RC5_decode () được thực thi khi cờ RC5FrameReceive bằng YES.

IR_RC5_DeInit ()

Chức năng này khởi tạo các thiết bị ngoại vi khác nhau (GPIO, TIMER, NVIC, ...).

TIM3_IRQHandler ()

Chức năng này xử lý ngắt TIM Capture Compare.

- **Sự kiện cạnh rơi bộ hẹn giờ:** Điều này được sử dụng để đo khoảng thời gian giữa hai cạnh rơi liên tiếp (toàn bộ khoảng thời gian xung).
- **Sự kiện Hẹn giờ Rising Edge:** Điều này được sử dụng để đo khoảng thời gian giữa các cạnh giảm và tăng (thời lượng xung thấp).
- **Cập nhật sự kiện (sự kiện hết giờ)** Điều này đặt lại gói RC5. Tràn bộ hẹn giờ được đặt thành 3,6 ms.

Khoảng thời gian xung thấp và toàn bộ thời lượng xung được sử dụng để xác định giá trị bit. Mỗi giá trị bit được xác định liên quan đến bit cuối cùng.

3.3.3**Cách sử dụng trình điều khiển bộ giải mã RC5**

Để sử dụng trình điều khiển bộ giải mã RC5, vui lòng thực hiện như sau:

- Gọi hàm IR_RC5_Init () để cấu hình bộ hẹn giờ và tài nguyên phần cứng GPIO cần thiết cho giải mã RC5.
- Các ngắt TIM3 Capture So sánh và Cập nhật được sử dụng để giải mã khung RC5, nếu một khung được nhận đúng, một biến toàn cục "RC5FrameReceive" sẽ được thiết lập để thông báo cho ứng dụng.
- Sau đó, ứng dụng sẽ gọi hàm IR_RC5_Decode () để lấy khung RC5 đã nhận.

Ví dụ về mã

```
# bao gồm "ir_decode.h"
# bao gồm <stdio.h>
# bao gồm "stm32100e_eval_lcd.h"
```

```
IR_Frame_TypeDef IR_FRAME;
/* Khởi tạo LCD màu gắn trên STM32100E-EVAL */ STM32100E_LCD_Init ();
```

```
/* Xóa màn hình LCD */
LCD_Clear (LCD_COLOR_WHITE);
IR_RC5_Init ();
/* RC5 */
IR_RC5_Decode (& IR_FRAME);
```

- Ghi chú:
- 1

TIM3_IRQHandler ISR được mã hóa trong trình điều khiển rc5_decode.

 - Nếu bạn đang sử dụng một hoặc cả hai ngắt trong ứng dụng của mình, bạn phải tiến hành cẩn thận:
 - Thêm mã ứng dụng của bạn vào các ISR này
 - Hoặc sao chép nội dung của các ISR này trong mã ứng dụng của bạn
- 2

Bạn có thể dễ dàng điều chỉnh ứng dụng này cho phù hợp với phần cứng của mình bằng cách sử dụng các khai báo xác định khác nhau bên trong tệp "rc5_decode.h". Vui lòng tham khảo Bảng sau.

Ban 2. Ví dụ về việc thực hiện

Xác định tên	Sự miêu tả	Giá trị được phép	Thí dụ
# xác định IR_TIM	Bộ hẹn giờ được sử dụng để giải mã IR	TIM1, TIM2, TIM3, TIM4, TIM5, TIM8, TIM9, TIM12, TIM15	TIM3
# định nghĩa TIM_PRESCALER	Bộ định mức TIM	Tham số này là tính toán theo cách có 1us làm cơ sở thời gian. 23 Tần số TIM (MHz) / (Bộ định mức + 1)	
# xác định IR_TIM_CLK	Đồng hồ APB của bộ đếm thời gian đã sử dụng	Tham khảo trình điều khiển RCC FW	RCC_APB1Periph_TIM3
# xác định IR_TIM_IRQn	IR TIM IRQ	Tham khảo stm32f10x_startup.s	TIM3_IRQn
# định nghĩa IR_TIM_Channel	Kênh IR TIM	TIM_Channel_1 hoặc TIM_Channel_2	TIM_Channel_1
# định nghĩa IR_GPIO_PORT	Cổng có đầu ra IR Tham khảo sản phẩm của bạn đã kết nối sơ đồ chân (1)		GPIOC
# định nghĩa IR_GPIO_PORT_CLK	Cổng đồng hồ GPIO chân hồng ngoại Tham khảo trình điều khiển RCC FW		RCC_APB2Periph_GPIOC
# xác định IR_GPIO_PIN	Ghim IR là kết nối	Tham khảo sơ đồ chân sản phẩm của bạn (1)	GPIO_Pin_6

1. Để biết thêm chi tiết về các tài nguyên STM32 hiện có, vui lòng tham khảo biểu dữ liệu sản phẩm của bạn.



3,4 So sánh các giải pháp RC5

Ghi chú ứng dụng này cung cấp hai giải pháp để triển khai bộ thu RC5 trong phần mềm, một giải pháp sử dụng EXTI và Bộ hẹn giờ đa năng (TIMx) và một giải pháp sử dụng Bộ hẹn giờ được định cấu hình ở chế độ Đầu vào PWM. Trình điều khiển này rất đơn giản để sử dụng và nó hỗ trợ các định dạng RC5 tiêu chuẩn và mở rộng, điều này không xảy ra với một số trình điều khiển RC5 do các nhà sản xuất khác cung cấp.

Bản số 3. Bảng so sánh

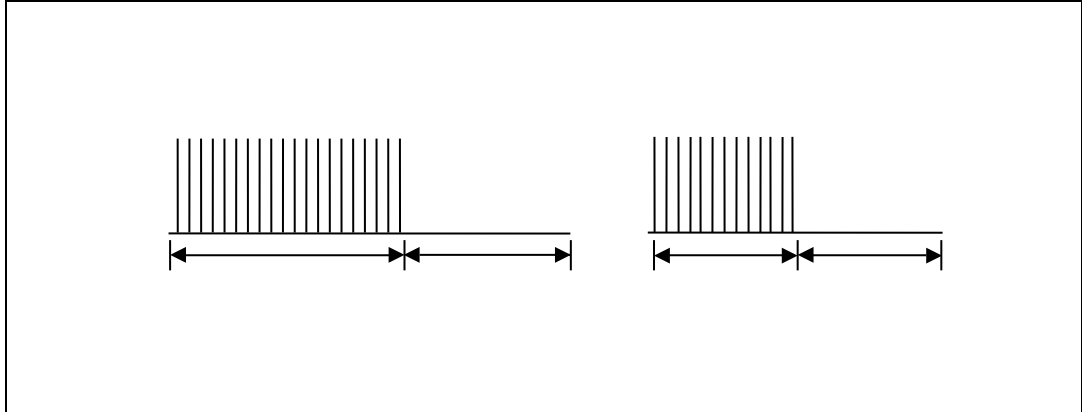
Giải pháp	Lợi thế	Hạn chế
Giải pháp đầu tiên: GPIO đơn với Bộ hẹn giờ chung	Mọi bộ hẹn giờ và GPIO đều có thể được sử dụng	Hai thiết bị ngoại vi STM32 được sử dụng (EXTI, TIMx)
Giải pháp thứ hai: Đầu vào PWM	Một thiết bị ngoại vi STM32 được sử dụng (TIMx)	TIMx phải có hai kênh và chỉ các GPIO cụ thể mới có thể được sử dụng

4 Giải pháp điều khiển hồng ngoại SIRC

4.1 Khái niệm cơ bản về giao thức

Mã SIRC là một từ 12 bit; nó sử dụng điều chế tần số sóng mang IR 40 kHz. Giao thức SIRC sử dụng mã hóa khoảng cách xung của các bit. Mỗi xung là một chùm sóng mang 40 kHz dài 600 μ s. Số logic "1" cần 1,8 ms để truyền, trong khi "0" logic mất 1,2 ms để truyền. (Tham khảo [Hình 11](#)).

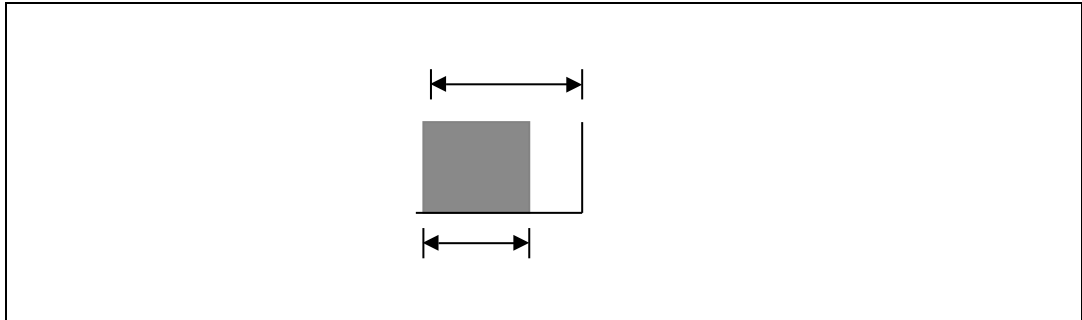
Hình 11. Độ dài của các bit logic



Khung SIRC chứa các trường sau:

- **Bit bắt đầu:** Khoảng bắt đầu luôn rộng 2,4 ms, theo sau là khoảng trống tiêu chuẩn là 0,6ms.
- **Độ dài 7 bit của lệnh:** Trường này chứa 7 bit được sử dụng làm trường lệnh.
- **Độ dài địa chỉ 5 bit:** Trường này chứa 5 bit được sử dụng làm trường địa chỉ.

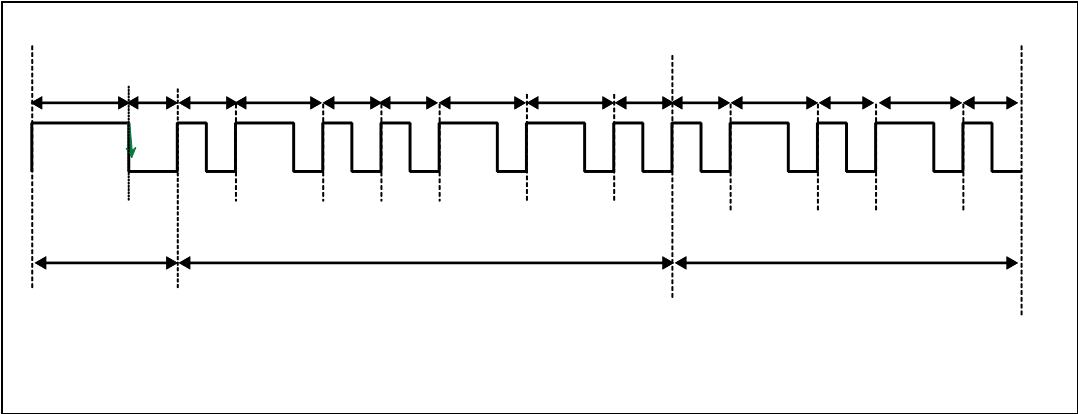
Hình 12. Độ dài của bit bắt đầu



Với giao thức này, LSB được truyền đầu tiên. Vì vậy, nó được lắp ráp LSB thành MSB. Vì nó được gửi dưới dạng 7 bit cho lệnh, tiếp theo là 5 bit cho địa chỉ thiết bị, mã phải chia 12 bit nhận được thành hai nhóm 7 và 5 bit.

Hình dưới đây cho thấy một ví dụ về khung SIRC.
 Trong trường hợp này: địa chỉ 26h (0100110b) và lệnh Ah (01010b)

Hình 13. Ví dụ về khung SIRC



Thời gian nhận rồi được chèn vào giữa hai khung liên tiếp để tránh va chạm. Cứ sau 45 ms, một mã lặp lại được truyền đi.

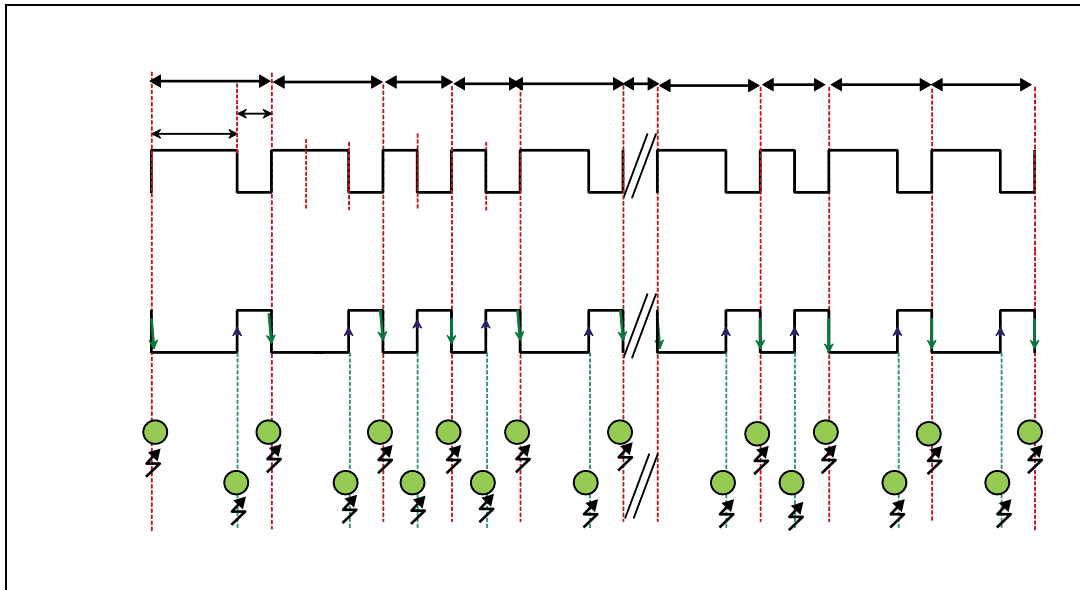
Bảng 4. SIRC thời gian

Sự miêu tả	Điện hình	Tối thiểu.	Tối đa
Syn xung mức cao Syn xung	2,4 mili giây	2,3 mili giây	2,6 mili giây
mức thấp bit 0 khoảng thời gian	0,6 mili giây	0,55 mili giây	0,7 mili giây
bit 1 period	1,2 mili giây	1,1 mili giây	1,3 mili giây
Thời gian nhận bản tin SIRC Thời gian bit xung	45 mili giây	-	-
của nhà cung cấp dịch vụ	25 μs	-	-

Ghi chú: *Bảng hiển thị tổng quan về dung sai độ rộng xung dữ liệu, được sử dụng trong ghi chú ứng dụng này. Người dùng có thể chỉ định thời gian SIRC tối thiểu-tối đa.*

4.2 Triển khai phần mềm

Hình 14. Cơ chế tiếp nhận khung SIRC

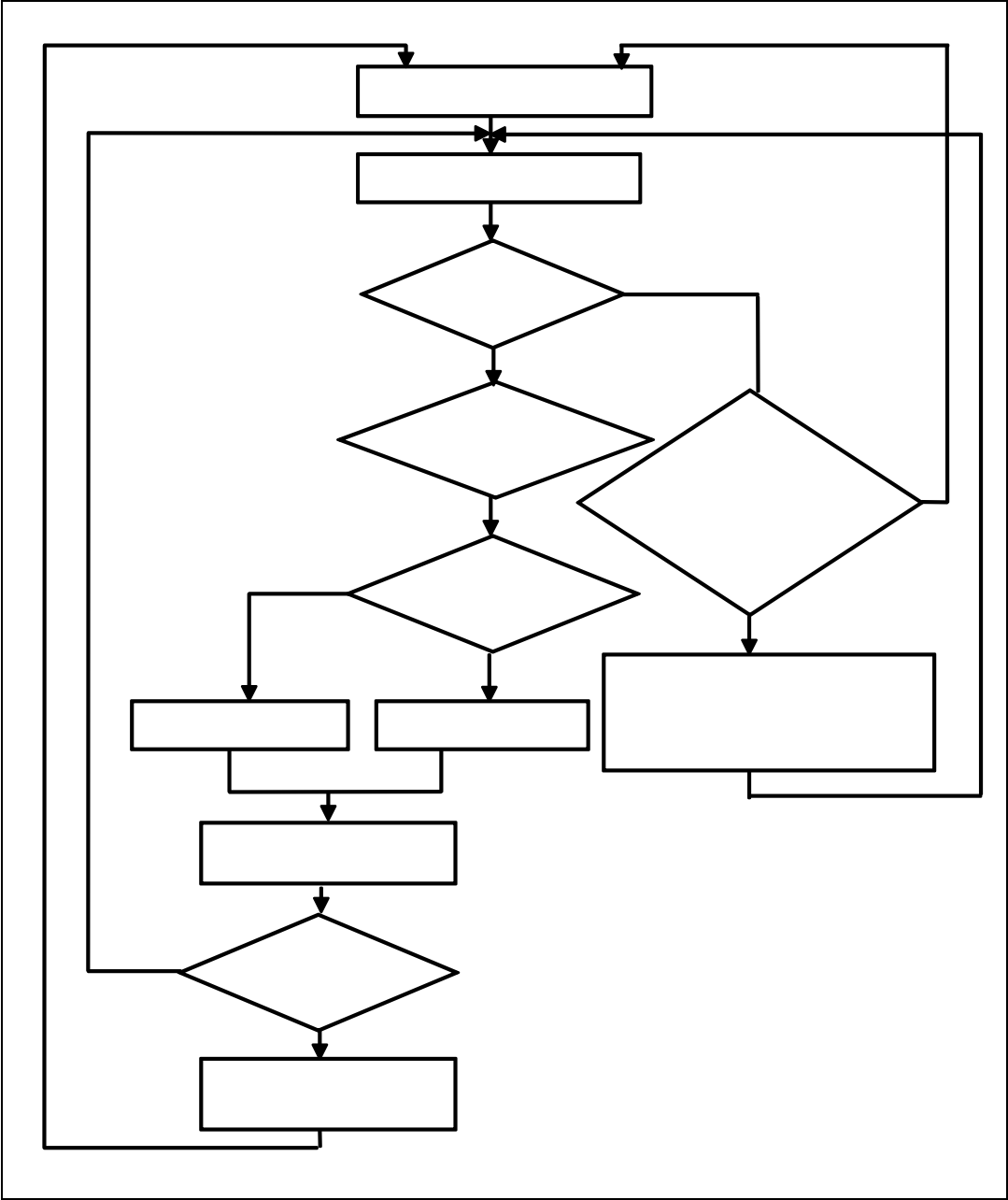


TIMER ngắt: ở chế độ đầu vào PWM

TIMER được sử dụng để lấy mẫu các bit khác nhau của khung SIRC. Chúng tôi nắm bắt giá trị bộ đếm thời gian hiện tại ở mỗi cạnh giảm và cạnh lên cũng như tạo ra một ngắt trên cả hai cạnh. Tính năng này giúp bạn dễ dàng đo toàn bộ xung SIRC và thời gian thấp.

- Nếu chu kỳ đo được bằng $T = 1200 \mu s$ và thời gian xung thấp bằng $T / 2 = 600 \mu s \Rightarrow$ bit là **logic '0'**.
- Nếu chu kỳ đo được bằng $3T / 2 = 1800 \mu s$ và thời gian xung thấp bằng $T = 1200 \mu s \Rightarrow$ bit là **logic '1'**.
- Nếu cả chu kỳ đo được bằng $3000 \mu s$ và khoảng thời gian xung thấp bằng $2400 \mu s \Rightarrow$ bit là **'start bit'**.

Hình 15. Lưu đồ giải pháp SIRC



4.3 Thư viện SIRC

IR_Init ()

Chức năng này khởi tạo các thiết bị ngoại vi khác nhau được sử dụng cho giao thức SIRC.

IR_Decode (IRFrame_TypeDef * ir_frame)

Chức năng này nhằm mục đích được gọi trong ứng dụng người dùng. Nó giải mã các tín hiệu SIRC nhận được. Nó có cấu trúc như một tham số chứa các giá trị khác nhau của khung IR.

```
typedef struct
{
    __IO uint8_t Địa chỉ;           /* Trường địa chỉ          */
    __IO uint8_t Lệnh;             /* Trường lệnh             */
} IRFrame_TypeDef;
```

IR_decode () phải được thực thi khi chờ IRFrameReceive bằng YES.

IR_ResetPacket ()

Chức năng này đặt gói IR về trạng thái mặc định. Hàm này được gọi trong quy trình TIM3_IRQHandler. Nó xảy ra mỗi lần tràn bộ đếm thời gian để đặt lại gói IR.

IR_DeInit ()

Chức năng này khởi tạo các thiết bị ngoại vi khác nhau được sử dụng cho giao thức SIRC.

TIM3_IRQHandler ()

Chức năng này xử lý ngắt TIM Capture Compare.

- **Sự kiện cạnh rơi bộ hẹn giờ:** Điều này được sử dụng để đo các chu kỳ khác nhau giữa hai cạnh rơi liên tiếp để xác định các bit khung.
- **Sự kiện Hẹn giờ Rising Edge:** Điều này được sử dụng để đo khoảng thời gian giữa các cạnh giảm và tăng (Thời lượng xung thấp).
- **Cập nhật sự kiện (sự kiện hết giờ)** Điều này đặt lại gói RC5. Tràn bộ hẹn giờ được đặt thành 4 ms.

Từ hai khoảng thời gian này, chúng tôi xác định giá trị bit.

4.4 Cách sử dụng trình điều khiển bộ giải mã SIRC

Để sử dụng trình điều khiển bộ giải mã SIRC, vui lòng thực hiện như sau:

- Các ngắt TIM3 Capture So sánh và Cập nhật được sử dụng để giải mã khung IR, nếu một khung được nhận đúng, một biến toàn cục "IRFrameReceive" sẽ được thiết lập để thông báo cho ứng dụng.
- Sau đó, ứng dụng sẽ gọi hàm IR_Decode () để lấy khung IR đã nhận.
- Bạn có thể dễ dàng điều chỉnh trình điều khiển này cho bất kỳ giao thức InfraRed nào khác, bằng cách điều chỉnh các định nghĩa từ sirc_decode.h thành đặc tả giao thức InfraRed (Bit Duration, Header Duration, Marge Tolerance, Number of bits ...) và các bảng lệnh và thiết bị từ ir_commands.c

Ví dụ về mã

```
# bao gồm "ir_decode.h"
# bao gồm <stdio.h>
# bao gồm "stm32100e_eval_lcd.h"
IR_Frame_TypeDef IR_FRAME;
/* Khởi tạo LCD màu gắn trên STM32100E-EVAL */ STM32100E_LCD_Init ();

/* Xóa màn hình LCD */
LCD_Clear (LCD_COLOR_WHITE);
IR_Init ();
/* SIRC */
IR_Decode (& IR_FRAME);
```

- Ghi chú:**
- 1 TIM3_IRQHandler ISR được mã hóa trong trình điều khiển ir_decode.c.**
 - Nếu bạn đang sử dụng một hoặc cả hai ngắt trong ứng dụng của mình, bạn phải tiến hành cẩn thận:
 - Thêm mã ứng dụng của bạn vào các ISR này
 - Hoặc sao chép nội dung của các ISR này trong mã ứng dụng của bạn
 - 2 Bạn có thể dễ dàng điều chỉnh ứng dụng này cho phù hợp với phần cứng của mình bằng cách sử dụng các khai báo xác định khác nhau bên trong tệp "ir_decode.h".**

Bảng 5. Ví dụ về việc thực hiện

Xác định tên	Sự miêu tả	Giá trị được phép	Thí dụ
# xác định IR_TIM	Bộ hẹn giờ được sử dụng để giải mã IR	TIM1, TIM2, TIM3, TIM4, TIM5, TIM8, TIM9, TIM12, TIM15	TIM3
# định nghĩa TIM_PRESCALER	Bộ định mức TIM	Tham số này là được tính toán theo cách để có 1us làm cơ sở thời gian. Tần số TIM (MHz) / (Bộ định mức + 1)	23
# xác định IR_TIM_CLK	Đồng hồ APB của bộ đếm thời gian đã sử dụng	Tham khảo trình điều khiển RCC FW	RCC_APB1Periph_TIM3

Xác định tên	Sự miêu tả	Giá trị được phép	Thí dụ
# xác định IR_TIM_IRQn	IR TIM IRQ	Tham khảo stm32f10x_startup.s	TIM3_IRQn
# định nghĩa IR_TIM_Channel	Kênh IR TIM	TIM_Channel_1 hoặc TIM_Channel_2	TIM_Channel_1
# định nghĩa IR_GPIO_PORT	Cổng có đầu ra IR Tham khảo sản phẩm của bạn đã kết nối sơ đồ chân (1)		GPIOC
# định nghĩa IR_GPIO_PORT_CLK	Cổng đồng hồ GPIO chân hồng ngoại Tham khảo trình điều khiển RCC FW		RCC_APB2Periph_GPIOC
# xác định IR_GPIO_PIN	Ghim IR là kết nối	Tham khảo sơ đồ chân sản phẩm của bạn (1)	GPIO_Pin_6

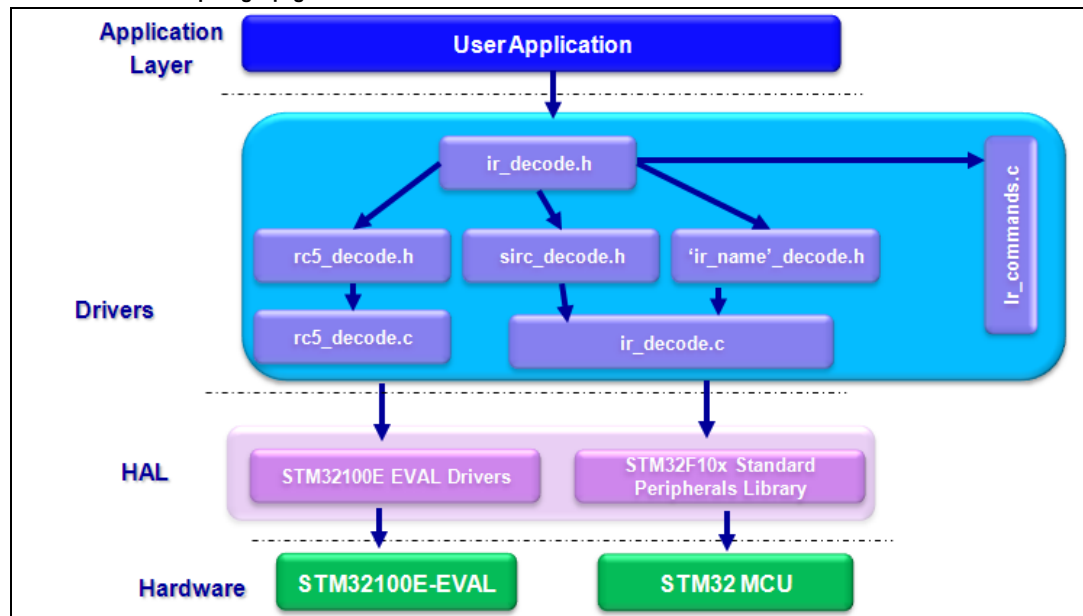
1. Để biết thêm chi tiết về các tài nguyên STM32 hiện có, vui lòng tham khảo biểu dữ liệu sản phẩm của bạn.



5 Lớp giao diện

Để tạo thuận lợi cho việc sử dụng các trình điều khiển giao thức hồng ngoại, một lớp giao diện (IR_decode) được sử dụng.

Hình 16. Kiến trúc lớp ứng dụng



Có thể lựa chọn Giao thức hồng ngoại thông qua định nghĩa phần sụn trong IR_decode.h.

```
# xác định IR_RC5_PROTOCOL
```

```
# xác định IR_SIRC_PROTOCOL
```

Có nhiều giao thức Hồng ngoại tương tự được phân biệt bằng các tham số thời gian như giao thức SIRC. Các giao thức này được xử lý bởi các hàm ir_decode.c. Bạn chỉ cần cập nhật các giá trị thời gian.

Có những cái khác khá khác biệt và được quản lý bởi các chức năng cụ thể như RC5 và trình điều khiển liên quan của nó rc5_decode.c

Mỗi giao thức có một khung cấu trúc cụ thể. IR_FRAME là một con trỏ đến cấu trúc giao thức hồng ngoại đã chọn và nó chứa thông tin chính cần thiết cho giao tiếp (DeviceAddress và Command).

Lớp này cung cấp một cách dễ dàng để sử dụng phần sụn.

```
int32_t main (void)
{
    # nếu được xác định IR_RC5_PROTOCOL
    IR_RC5_Init ();
    # khác
    IR_Init ();
    # endif

    trong khi (1)
    {
        # nếu được xác định (IR_RC5_PROTOCOL)
        /* RC5 */
        IR_RC5_Decode (& IR_FRAME);
        # khác
        IR_Decode (& IR_FRAME);
        # endif
    }
}
```

6 Cách sử dụng trình điều khiển IR

Để chạy các ví dụ về ứng dụng InfraRed, vui lòng tiến hành như sau:

- Mở dự án có tên "IR_Decoding_PWM" (đường dẫn: \ Project \ InfraRed \ IR_Decoding_PWM)
- Chọn chuỗi công cụ ưa thích của bạn
- Chọn không gian làm việc tương ứng trong dự án chuỗi công cụ ưa thích của bạn. Bạn có thể chọn IR_RC5_Decode hoặc IR_SIRC_Decode. Các không gian làm việc này đã được định cấu hình.

6.1 Các chương trình trình diễn

6.1.1 Demo sử dụng GP-Timer được định cấu hình ở chế độ PWM

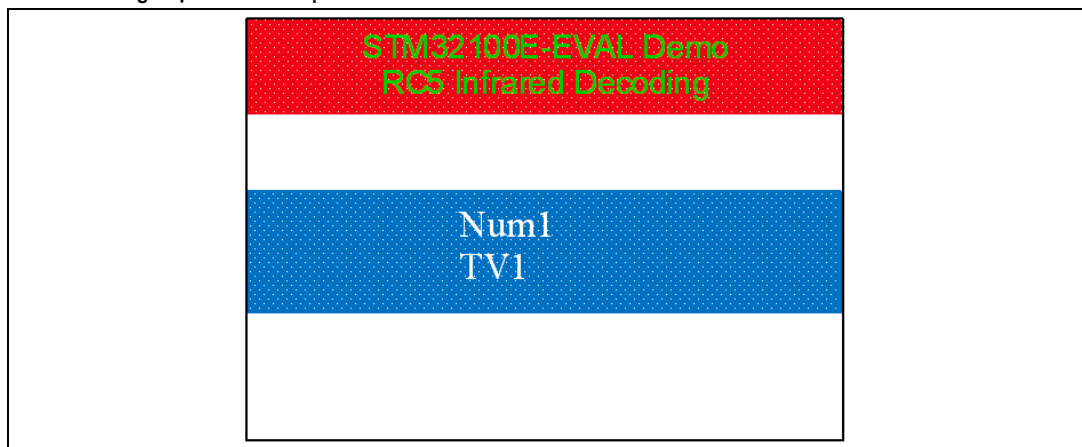
Bản demo này bao gồm việc nhận các thông điệp IR và gửi chúng đến màn hình LCD.

Mỗi thông báo IR được hiển thị thành 2 phần:

- Thiết bị truyền khung IR Lệnh được thực thi
-

Hình sau cho thấy Bộ giải mã RC5 sử dụng phương pháp PWM ([Phần 3.3: Triển khai phần mềm bằng cách sử dụng GP-Timer được định cấu hình ở chế độ đầu vào PWM trên trang 10](#)).

Hình 17. Khung nhận RC5 hiển thị trên màn hình LCD



6.1.2 Bản demo RC5 sử dụng một GPIO duy nhất với bộ đếm thời gian chung

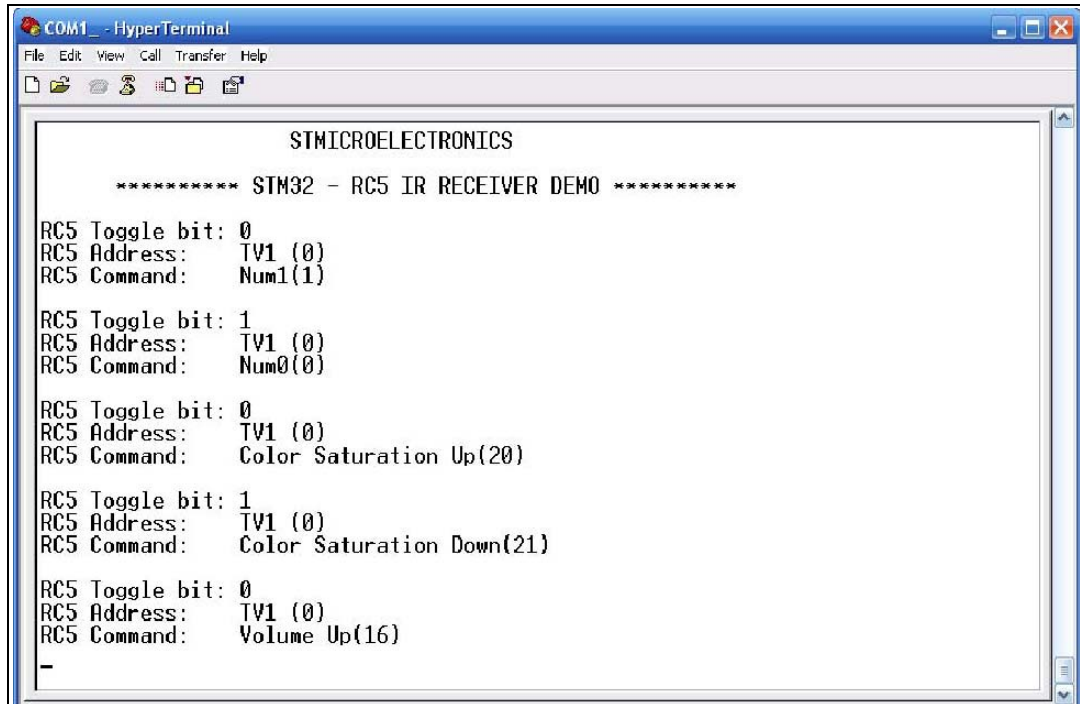
Bản trình diễn này bao gồm việc nhận các thông điệp RC5 và gửi chúng đến hyperterminal (xem [Hình 18](#)) sử dụng USART1 (115200 baud, dữ liệu 8 bit, Không chặn lẻ, Không kiểm soát luồng). Mỗi thông báo RC5 được hiển thị thành 3 phần:

- Giá trị của bit chuyển đổi.
- Thiết bị đã truyền RC5 với giá trị thập phân của nó trong dấu ngoặc. Lệnh sẽ được thực thi và giá trị thập phân của nó trong dấu ngoặc.

Khi nhận được thông báo RC5, LED1 sẽ bật tắt trên bảng.

Xem thêm [Phần 3.2: Triển khai phần mềm bằng cách sử dụng một GPIO duy nhất với bộ đếm thời gian chung trên trang 7](#)).

Hình 18. Các khung nhận RC5 được hiển thị trong hyperterminal (giải pháp hẹn giờ cơ bản)



```
STMICROELECTRONICS
***** STM32 - RC5 IR RECEIVER DEMO *****

RC5 Toggle bit: 0
RC5 Address:   TV1 (0)
RC5 Command:   Num1(1)

RC5 Toggle bit: 1
RC5 Address:   TV1 (0)
RC5 Command:   Num0(0)

RC5 Toggle bit: 0
RC5 Address:   TV1 (0)
RC5 Command:   Color Saturation Up(20)

RC5 Toggle bit: 1
RC5 Address:   TV1 (0)
RC5 Command:   Color Saturation Down(21)

RC5 Toggle bit: 0
RC5 Address:   TV1 (0)
RC5 Command:   Volume Up(16)
-
```

7 Cách tùy chỉnh trình điều khiển IR

Để bao gồm trình điều khiển Hồng ngoại dựa trên giải pháp đầu vào PWM trong ứng dụng người dùng, bạn phải:

1. Mở dự án có tên "IR_Decoding_PWM1"
2. Thêm tệp tiêu đề của giao thức IR thích hợp vào dự án của bạn. Ví dụ: rc5_decode.h.
3. Thêm tệp.c tương ứng với giao thức IR vào dự án của bạn. Ví dụ: rc5_decode.c.
4. Gọi hàm khởi tạo giao thức trong hàm main ()
Thí dụ: IR_RC5_Init ();
5. Thêm vào stm32f10x_it.c: hàm ngắt TIMx.

Thí dụ:

```
void TIM3_IRQHandler (void)
{
    uint32_t tính ICValue1;
    uint32_t tính ICValue2;
    /* Ngắt IC1 */
    if ((TIM_GetFlagStatus (IR_TIM, TIM_FLAG_CC1) != RESET)) {

        TIM_ClearFlag (IR_TIM, TIM_FLAG_CC1);
        /* Nhận giá trị Ghi đầu vào */ ICValue2 = TIM_GetCapture1
        (IR_TIM);
        /* RC5 */
        IR_RC5_DataSampling (ICValue2 - ICValue1, 0); }

    /* Ngắt IC2 */
    else if ((TIM_GetFlagStatus (IR_TIM, TIM_FLAG_CC2) != RESET)) {

        TIM_ClearFlag (IR_TIM, TIM_FLAG_CC2);
        /* Nhận giá trị Ghi đầu vào */ ICValue1 = TIM_GetCapture2
        (IR_TIM);
        IR_RC5_DataSampling (ICValue1, 1);
    }
    /* Kiểm tra xem cờ IR_TIM đã được đặt hay chưa. */
    else if ((TIM_GetFlagStatus (IR_TIM, TIM_FLAG_Update) != RESET)) {

        /* Xóa các cờ đang chờ xử lý của IR_TIM */ TIM_ClearFlag (IR_TIM,
        TIM_FLAG_Update);
        IR_RC5_ResetPacket ();
    }
}
```

6. Xác định cấu trúc cho giao thức IR trong tệp main.c. Thí dụ:

IRFrame_TypeDef IR_FRAME.

7. Gọi hàm giải mã trong hàm main (). Thí dụ:

```
void main (void)
{
    IR_RC5_Init ();
    trong khi (1)
    {
        IR_RC5_Decode (& IR_Frame).
    }
}
```

Đối với giải pháp bộ giải mã RC5 trên GPIO với bộ đếm thời gian chung, bạn phải:

1. Mở dự án có tên 'RC5_Decoding_TIM_EXTI'
2. Thêm tệp tiêu đề của giao thức RC5 vào dự án của bạn: RC5_IR_Emul_Receiver.h
3. Thêm tệp nguồn vào dự án của bạn: RC5_IR_Emul_Receiver.c
4. Xác định một hình ảnh vũng chắc cho RC5.

Thí dụ:

RC5Frame_TypeDef RC5_FRAME.

5. Thêm tệp stm32f10x_it.c có chứa ngắt TIMx và các hàm EXTI. Gọi hàm giải mã
- 6.

Thí dụ:

```
void main (void)
{
    RC5_Receiver_Init ();
    trong khi (1)
    {
        if (RC5_Frame_Receive == YES)
        {
            RC5_FRAME = IR_RC5_Decode ().
        }
    }
}
```

Các thay đổi cần thiết để hỗ trợ bất kỳ giao thức IR nào

Bạn có thể sử dụng giải pháp này để hỗ trợ bất kỳ giao thức Hồng ngoại nào bằng cách chỉ thực hiện một số thay đổi trong tệp tiêu đề và cập nhật lệnh và bảng thiết bị.

- Tạo một tệp tiêu đề (exp: ir_protocol_name.h) tương tự như tệp sinc_decode.h. Thay đổi các định nghĩa để điều chỉnh nó cho phù hợp với các thông số kỹ thuật của giao thức IR đã chọn (Thời lượng bit Tối thiểu / Tối đa, Thời lượng tiêu đề Tối thiểu / Tối đa, Tổng số bit, Thời gian kết thúc ...)

Bảng 6. Danh sách các định nghĩa trong tệp tiêu đề cho các tham số giao thức IR

Xác định	Ý nghĩa	Cài đặt ví dụ cho Giao thức SIRC
IR_Time_OUT_US	Thời gian chờ tính bằng μ s	4000
IR_BITS_COUNT	Số lượng bit	11
IR_TOTAL_BITS_COUNT	Tổng số bit	11
IR_ONTIME_MIN_US	Min Xung thấp tính bằng μ s	(600 - 50)
IR_ONTIME_MAX_US	Xung thấp tối đa tính bằng μ s	(1200 + 50)
IR_HEADER_LOW_MIN_US	Tiêu đề tối thiểu Xung thấp tính bằng μ s	(2400 - 100)
IR_HEADER_LOW_MAX_US	Tiêu đề tối đa Xung thấp tính bằng μ s	(2400 + 100)
IR_HEADER_WHOLE_MIN_US	Thời lượng toàn bộ Tiêu đề tối thiểu tính bằng μ s	(2400 + 600 - 50)
IR_HEADER_WHOLE_MAX_US	Toàn bộ thời lượng Tiêu đề tối đa tính bằng μ s	(2400 + 600 + 50)
IR_VALUE_STEP_US	Giá trị bước giữa bit0 và bit1 tính bằng μ s	600
IR_VALUE_MARGIN_US	Lợi nhuận tính bằng μ s	100
IR_VALUE_00_US	Thời lượng bit0 tính bằng μ s	1200

- Thay đổi trường khung giao thức IR trong cấu trúc IR_Frame_TypeDef.

```
typedef struct
{
    /* Cấu trúc của khung IR (Địa chỉ, Lệnh, ....) */

} IR_Frame_TypeDef;
```
- trong tệp ir_commands.c, thêm các bảng IR_Commands và IR_devices thích hợp cho giao thức IR.
- trong tệp ir_decode.c, hãy cập nhật hàm IR_Decode (IR_Frame_TypeDef * ir_frame) với cấu trúc khung của giao thức đích.

số 8 Phản kết luận

Ghi chú ứng dụng này cung cấp giải pháp triển khai bộ thu IR trong phần mềm bằng cách sử dụng bộ hẹn giờ mục đích chung.

Trình điều khiển IR cho phép tích hợp giải pháp IR trong mô-đun HDMI-CEC để hỗ trợ các chức năng điều khiển cấp cao cho tất cả các sản phẩm nghe nhìn khác nhau trong một môi trường nhất định.

9 Lịch sử sửa đổi

Bảng 7. Lịch sử sửa đổi tài liệu

Ngày	Bản sửa đổi	Những thay đổi
01-04-2010	1	Phát hành lần đầu.
17-02-2012	2	Được cập nhật với giải pháp chung để hỗ trợ các giao thức IR khác nhau

Vui lòng đọc kỹ:

Thông tin trong tài liệu này chỉ được cung cấp liên quan đến các sản phẩm ST. STMicroelectronics NV và các công ty con của nó ("ST") bảo lưu quyền thực hiện các thay đổi, hiệu chỉnh, sửa đổi hoặc cải tiến đối với tài liệu này cũng như các sản phẩm và dịch vụ được mô tả ở đây vào bất kỳ lúc nào mà không cần thông báo.

Tất cả các sản phẩm của ST được bán theo các điều khoản và điều kiện bán hàng của ST.

Người mua hoàn toàn chịu trách nhiệm về việc lựa chọn, lựa chọn và sử dụng các sản phẩm và dịch vụ của ST được mô tả ở đây và ST không chịu bất kỳ trách nhiệm pháp lý nào liên quan đến việc lựa chọn, lựa chọn hoặc sử dụng các sản phẩm và dịch vụ của ST được mô tả ở đây.

Không có giấy phép, rõ ràng hay ngụ ý, bởi estoppel hoặc cách khác, đối với bất kỳ quyền sở hữu trí tuệ nào được cấp theo tài liệu này. Nếu bất kỳ phần nào của tài liệu này đề cập đến bất kỳ sản phẩm hoặc dịch vụ nào của bên thứ ba, nó sẽ không được ST coi là cấp giấy phép cho việc sử dụng các sản phẩm hoặc dịch vụ của bên thứ ba đó, hoặc bất kỳ tài sản trí tuệ nào có trong đó hoặc được coi là bảo hành bao gồm việc sử dụng trong bất kỳ hình thức nào của các sản phẩm hoặc dịch vụ của bên thứ ba đó hoặc bất kỳ tài sản trí tuệ nào có trong đó.

KHÔNG CÓ LỜI NÓI KHÁC ĐƯỢC ĐẶT RA TRONG ĐIỀU KHOẢN VÀ ĐIỀU KIỆN BÁN ST TỪ CHỐI BẤT CỨ BẢO ĐẢM RÕ RÀNG HOẶC NGỤ Ý LIÊN QUAN ĐẾN VIỆC SỬ DỤNG VÀ / HOẶC BÁN SẢN PHẨM ST BAO GỒM KHÔNG GIỚI HẠN BẢO ĐẢM NGUỒN LỰC VỀ KHẢ NĂNG BẢO ĐẢM, HỢP PHÁP VÀ MỤC ĐÍCH CỦA CHÚNG TÔI (CỦA BẤT KỲ LUẬT PHÁP NÀO), HOẶC XÂM PHẠM BẤT KỲ BẤT KỲ BẤT KỲ BẢN QUYỀN NÀO, BẢN QUYỀN NÀO HOẶC QUYỀN SỞ HỮU TRÍ TUỆ KHÁC.

KHÔNG ĐƯỢC PHÉP DUYỆT RÕ RÀNG TRONG BÀI VIẾT CỦA HAI NGƯỜI ĐẠI DIỆN ĐƯỢC ỦY QUYỀN CỦA ST, CÁC SẢN PHẨM CỦA ST KHÔNG ĐƯỢC ĐỀ XUẤT, ĐƯỢC ỦY QUYỀN BẢO ĐẢM NGAY LẬP TỨC, MÂY BAY, KHÔNG GIAN, TIẾT KIỆM CUỘC SỐNG, HOẶC CÁC ỨNG DỤNG ĐÚNG ĐỜI SỐNG, BẰNG CÁCH NÓI TIẾP TRONG HỆ THỐNG SẢN PHẨM HOẶC CHẾT HOẶC THIẾT HẠI VỀ TÀI SẢN HOẶC MÔI TRƯỜNG. CÁC SẢN PHẨM ST KHÔNG ĐƯỢC QUY ĐỊNH CỤ THỂ LÀ "LỚP Ồ TỎ" CHỈ CÓ THỂ ĐƯỢC SỬ DỤNG TRONG CÁC ỨNG DỤNG Ồ TỎ VỚI RỦI RO RIÊNG CỦA NGƯỜI DÙNG.

Việc bán lại các sản phẩm ST với các điều khoản khác với các tuyên bố và / hoặc các tính năng kỹ thuật được nêu trong tài liệu này sẽ ngay lập tức làm mất hiệu lực của bất kỳ bảo hành nào được ST cấp cho sản phẩm hoặc dịch vụ ST được mô tả ở đây và sẽ không tạo ra hoặc mở rộng theo bất kỳ cách nào, bất kỳ trách nhiệm nào của ST.

ST và biểu tượng ST là nhãn hiệu hoặc nhãn hiệu đã đăng ký của ST ở các quốc gia khác nhau.

Thông tin trong tài liệu này thay thế và thay thế tất cả thông tin đã cung cấp trước đó.

Biểu trưng ST là nhãn hiệu đã đăng ký của STMicroelectronics. Tất cả các tên khác là tài sản của chủ sở hữu tương ứng của họ.

© 2012 STMicroelectronics - Mọi quyền được bảo lưu

Nhóm công ty STMicroelectronics

Úc - Bỉ - Brazil - Canada - Trung Quốc - Cộng hòa Séc - Phần Lan - Pháp - Đức - Hồng Kông - Ấn Độ - Israel - Ý - Nhật Bản -
Malaysia - Malta - Maroc - Philippines - Singapore - Tây Ban Nha - Thụy Điển - Thụy Sĩ - Vương quốc Anh - Hợp chúng quốc Hoa Kỳ

www.st.com