Chương 5

NGẮT VÀ SỰ KIỆN

INTERRUPTS AND EVENTS

5.1 BỘ ĐIỀU KHIỂN VECTOR NGẮT LỒNG NHAU

(NVIC-Nested Vector Interrupt Controller)

5.1.1 Ngắt là gì ?

Ngắt là việc vi điều khiển dừng tạm thời chương trình đang thực thi mà đi thực thi một chương trình khác có yêu cầu cấp thiết hơn. Sau khi thực thi xong CPU sẽ quay trở về thực hiện lại chương trình cũ.

Ví dụ: khi điều khiển màn hình cảm ứng ta không nên bắt vi điều khiển phải kiểm tra liên tục việc người dùng có chạm vào màn hình hay không vì nếu làm vậy sẽ gây giảm tốc độ thực thi các tác vụ khác trong khi người dùng thì không phải lúc nào cũng thao tác với màn hình. Đối với trường hợp này ta nên **sử dụng ngắt** có nghĩa là lúc này CPU sẽ giao việc kiểm tra màn hình cho các khối khác là EXTI và NVIC quản lý và nhờ vậy CPU có nhiều thời gian để xử lý các tác vụ khác hơn. Khi người dùng chạm vào màn hình thì EXTI sẽ phát ra yêu cầu ngắt và yêu cầu này được gởi tới NVIC để NVIC sắp xếp yêu cầu CPU tạm dừng chương trình đang thực hiện lại và đi kiểm tra xem người dùng đang nhấn chỗ nào trên màn hình để giải quyết yêu cầu tương ứng với vị trí nhấn đó. Sau khi thực thi xong các yêu cầu của phím nhấn CPU sẽ trở về làm tiếp việc hiện tại.

5.1.2 NVIC là gì ?

NVIC là một khối nằm trong lõi Cortex có nhiệm vụ quản lý các nguồn ngắt, do có rất nhiều nguồn có khả năng yêu cầu CPU ngắt nên chúng cần có một **"người quản lý"** chung để sắp xếp, điều khiển mọi hoạt động ngắt được diễn ra đúng theo ý muốn tránh trường hợp chồng chéo, xung đột giữa các yêu cầu.

NVIC là một khối chuẩn trong lõi Cortex. Điều này có nghĩa là tất cả các vi điều khiển dựa trên lỗi Cortex sẽ có cùng một cấu trúc ngắt bất kể nhà sản xuất chip Atmel, Ti , ST... Nhờ vậy mà người lập trình có thể kế thừa chương trình của của hãng này cho chip hãng khác.

NVIC được thiết kế để điều khiển các nguồn ngắt lồng nhau và trên STM32 NVIC có:

- 16 cấp độ ưu tiên ngắt.
- 68 kênh ngắt có thể che được(maskable –có thể cho phép hoặc cấm được) – Chưa kể đến 16 đường ngắt của lõi Cortex™-M3.

5.2 CƠ CHẾ ƯU TIÊN NGẮT

Khi có nhiều nguồn ngắt xảy ra cùng một lúc thì CPU sẽ xử lý yêu cầu của ngắt có độ ưu tiên cao nhất dần dần cho đến thấp nhất. Chỉ trừ một số ngắt không thể che thuộc về phần cứng thì thứ tự ưu tiên của các nguồn ngắt còn lại có thể được quy định bởi người lập trình.

Có 16 cấp độ ưu tiên ngắt(được điều khiển bằng **4 bit**) và 4 bit này có thể được phân chia theo 5 tỉ lệ giữa nhóm ưu tiên chính (PreemptionPriority) và nhóm ưu tiên phụ (SubPriority) theo **bảng 5.1.**

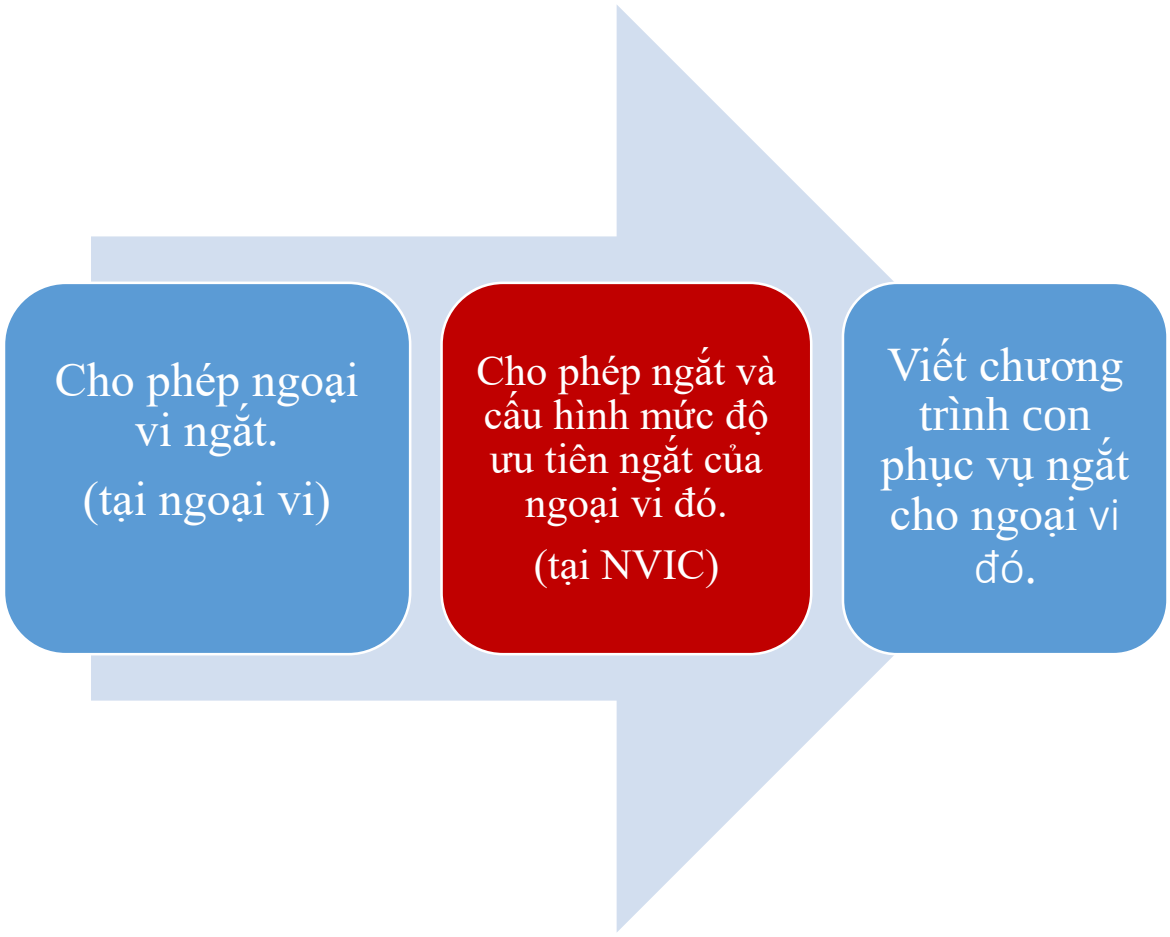
Bảng 5.1 Bảng phân chia số cấp ưu tiên giữa 2 nhóm PreemptionPriority và SubPriority

Cách phân chia		PreemptionPriority		SubPriority	
Tỉ lệ số bit Preem: Sub	Lệnh trong lập trình NVIC_PriorityGroupConfig(**A**);	Số cấp ưu tiên	Phạm vi	Số cấp ưu tiên	Phạm vi
0 bit : 4 bit	**A**= NVIC_PriorityGroup_0	0	0	16	0-15
1 bit : 3 bit	**A**=NVIC_PriorityGroup_1	2	0-1	8	0-7
2 bit : 2 bit	**A**=NVIC_PriorityGroup_2	4	0-3	4	0-3
3 bit : 1 bit	**A**=NVIC_PriorityGroup_3	8	0-7	2	0-1
4 bit : 0 bit	**A**=NVIC_PriorityGroup_4	16	0-15	0	0

Chú ý : Nhóm PreemptionPriority có mức độ ưu tiên cao hơn SubPriority và cấp độ ưu tiên càng nhỏ thì ưu tiên càng cao. Có nghĩa là cấp 0 của nhóm PreemptionPriority sẽ có mức độ ưu tiên cao nhất.

5.3 CÁC BƯỚC ĐỂ CẤU HÌNH SỬ DỤNG NGẮT

Để có thể sử dụng được ngắt của 1 ngoại vi nào đó ta làm theo 3 bước sau:



Hình 5.1 Các bước để sử dụng ngắt

5.4 CÁC LỆNH THÔNG DỤNG LIÊN QUAN ĐẾN NVIC

Bảng 5.2 Các lệnh thông dụng liên quan đến NVIC

SỬ DỤNG THƯ VIỆN "misc"	
Lệnh	
Thông số hay dùng	**Giải thích**
NVIC_SetVectorTable(A,B); (Lệnh chọn nơi lưu bảng vector ngắt)	
A: NVIC_VectTab_RAM NVIC_VectTab_FLASH	**A:** Nơi lưu Lưu trong RAM Lưu trong FLASH **B:** Địa chỉ đầu vùng nhớ cần lưu. Địa chỉ này phải là bội số của 0x200
NVIC_PriorityGroupConfig(A); (Lệnh chọn cách phân chia số cấp ưu tiên của 2 nhóm PreemptionPriority và SubPriority)	
A: NVIC_PriorityGroup_0 NVIC_PriorityGroup_1 NVIC_PriorityGroup_2 NVIC_PriorityGroup_3 NVIC_PriorityGroup_4	**A:** Nhóm ưu tiên được chọn Nhóm 0: 0 Pre – 16 Sub Nhóm 1: 2 Pre – 8 Sub Nhóm 2: 4 Pre – 4 Sub Nhóm 3: 8 Pre – 2 Sub Nhóm 4: 16 Pre – 0 Sub **(Xem bảng 5.1)**
NVIC_InitTypeDef A; **A.NVIC_IRQChannel = B;** (Lệnh chọn ngoại vi cần cho phép ngắt)	
B: USART1_IRQn SPI1_IRQn EXTI4_IRQn **…** TIM2_IRQn	**B:** Ngoại vi cần ngắt Ngắt UART1 Ngắt SPI1 Ngắt ngoài kênh số 4 **…** Ngắt Timer2 **Chú ý:** Xem trong file **"stm32f10x.h"** từ dòng 169 đến 470 để biết cách viết tên ngoại vi cần cho phép ngắt.
A.NVIC_IRQChannelPreemptionPriority = C; (Lệnh chọn thứ tự ưu tiên ngắt của ngoại vi đang cấu hình trong nhóm chính)	
C: 0 … 15	**C:** thứ tự ưu tiên 0: Ưu tiên cao nhất … 15: Ưu tiên thấp nhất

A.NVIC_IRQChannelSubPriority = D; (Lệnh chọn thứ tự ưu tiên ngắt của ngoại vi đang cấu hình trong nhóm phụ)	
D: 0 … 15	**D:** thứ tự ưu tiên 0: Ưu tiên cao nhất … 15: Ưu tiên thấp nhất
A.NVIC_IRQChannelCmd = E; (Lệnh cho phép hoặc cấm ngắt ngoại vi đang được cấu hình tại NVIC)	
E: ENABLE DISABLE	**E:** Cho phép hoặc cấm Cho phép ngắt Cấm ngắt
NVIC_Init(&A); (Lệnh cấu hình các thông số được lưu trong biến A cho NVIC)	

5.5 CÁC VÍ DỤ LIÊN QUAN ĐẾN NVIC

Ví dụ 5.1: Viết chương trình con cấu hình NVIC cho phép ngắt ngoài kênh số 4

Chương trình

```
void CauhinhNVIC_EXTI4()
{
    #ifdef  VECT_TAB_RAM
      NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
      // Nếu có định nghĩa "VECT_TAB_RAM" thì
      // lưu bảng vector ngắt vào RAM
    #else
      NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
      // Nếu không định nghĩa "VECT_TAB_RAM"
      // thì lưu bảng vector ngắt vào FLASH
    #endif
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
    // 0 cấp PreemptionPriority và 16 cấp SubPriority
    NVIC_InitStructure.NVIC_IRQChannel = EXTI4_IRQn;
    // Chọn ngoại vi cần ngắt là ngắt ngoài kênh 4
    //NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    //Bỏ dòng trên vì chọn chế độ PreemPtionPriority có 0 cấp
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    // Chọn thứ tự ưu tiên 1 trong SubPriority cho kênh 4
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    // Cho phép ngắt
    NVIC_Init(&NVIC_InitStructure);
    // Cài đặt các cấu hình trên vào NVIC
  }
```

Chú ý: Khi đã cấu hình NVIC cho phép ngoại vi ngắt thì ta phải viết chương trình con phục vụ ngắt cho ngoại vi đó. Cấu trúc của chương trình con phục vụ ngắt giống với cấu trúc của một chương trình con bình thường và chúng chỉ khác nhau ở phần tên. Có nghĩa là muốn viết chương trình con phục vụ ngắt ta viết chương trình con như bình thường và đặt tên cho nó đúng với cú pháp sau:

Tên ngoại vi khai báo ngắt trong NVIC **bỏ đi** chữ **"n"** ở cuối **thêm vào** chữ **"Handler"**

Như **ví dụ 5.1** trên ta sẽ đặt tên chương trình con phục vụ ngắt là:

`EXTI4_IRQn` bỏ đi chữ **"n"** ở cuối **thêm vào** chữ **"Handler"** = EXTI4_IRQHandler

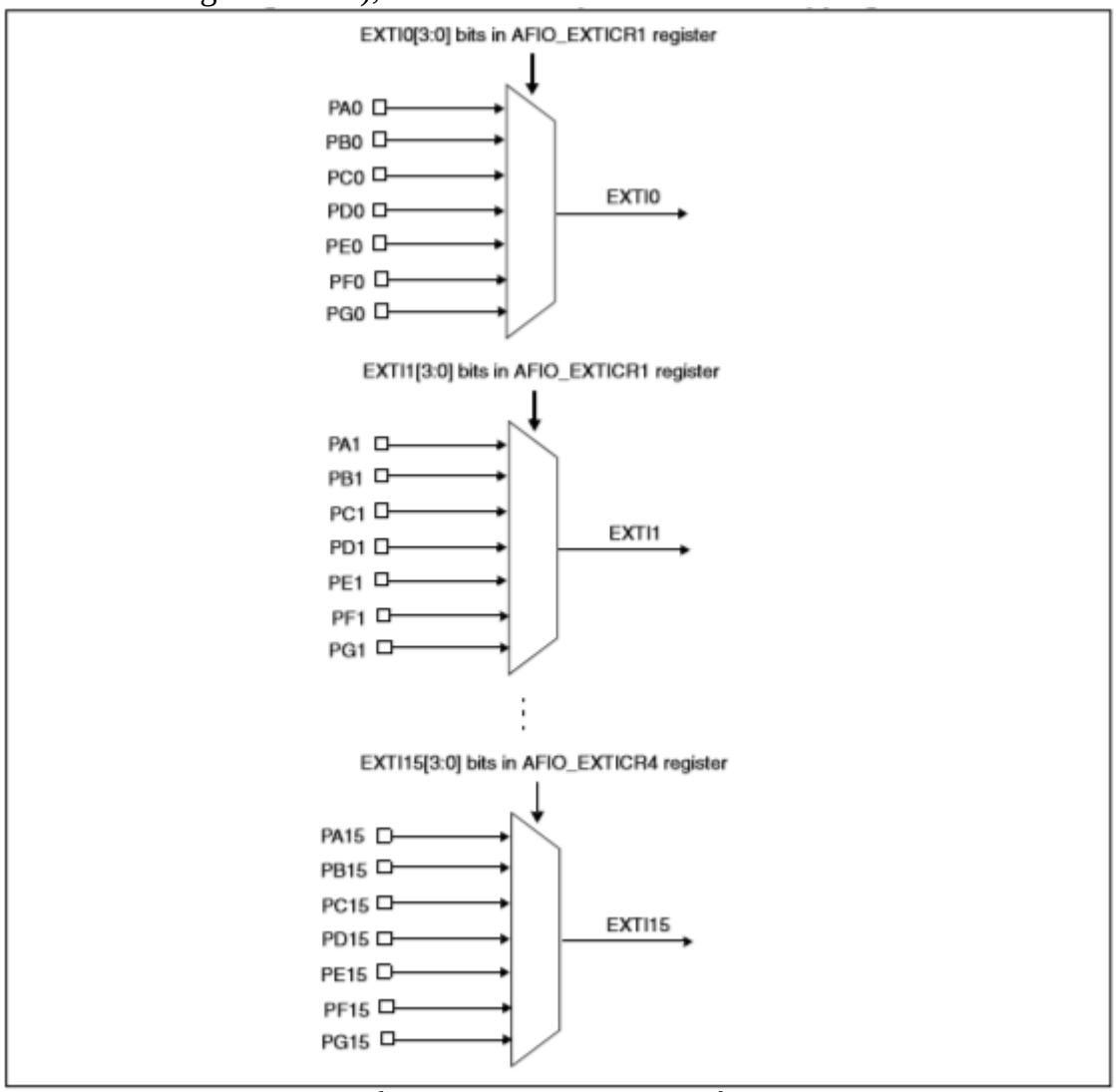
```
void  EXTI4_IRQHandler (void)
{
    // Các lệnh thực thi kh ngắt ngoài xảy ra viết ở đây
}
```

5.6 NGẮT NGOÀI(EXTERNAL INTERRUPT – EXTI)

5.6.1 Giới thiệu về EXTI

Hầu hết tất cả các chân ARM của ST đều có thể thiết lập ngắt ngoài chỉ ngoại từ một số chân được biệt .

Có 19 kênh [EXT0 – EXT18] phục vụ cho ngắt ngoài. Trong đó các kênh từ 0 đến 15 phục vụ yêu ngắt ngoài tương ứng với các chân từ 0 đến 15 của các PORT, kênh 16 nối với PVD (Programmable Voltage Detector), kênh 17 kết nối RTC Alarm và kênh 18 là USB Wakeup.



Hình 5.2 bảng đồ GPIO của các kênh ngắt ngoài và sự kiện

Mỗi kênh có thể được cấu hình riêng để có thể phát hiện xung hoặc cạnh (cạnh lên, cạnh xuống, cả cạnh lên lẫn cạnh xuống), cho phép hoặc cấm ngắt.

5.6.2 Các lệnh thông dụng liên quan đến ngắt ngoài và sự kiện

Bảng 5.3 Các lệnh thông dụng liên quan đến ngắt ngoài và sự kiện

SỬ DỤNG THƯ VIỆN "stm32f10x_exti"	
Lệnh	
Thông số hay dùng	**Giải thích**
EXTI_InitTypeDef A; **GPIO_EXTILineConfig(B, C);** (Lệnh chọn chân nhận tín hiệu ngắt ngoài/sự kiện)	
B: GPIO_PortSourceGPIOA … GPIO_PortSourceGPIOG **C:** GPIO_PinSource0 … GPIO_PinSource15	**B:** Chọn PORT Chọn PORTA … Chọn PORTG **C:** Chọn chân Chọn chân 0 … Chọn chân 15
A.EXTI_Line = D; (Lệnh chọn kênh ngắt ngoài/sự kiện)	
D: EXTI_Line0 … EXTI_Line18	**D:** Kênh cần chọn Chọn kênh 0 … Chọn kênh 18
A.EXTI_Mode = E ; (Lệnh chọn chế độ ngắt hoặc sự kiện)	
E: EXTI_Mode_Interrupt EXTI_Mode_Event	**E:** Chọn chế độ Chế độ ngắt ngoài Chế độ tạo sự kiện
A.EXTI_Trigger = F; (Lệnh chọn cạnh tích cực)	
F: EXTI_Trigger_Rising EXTI_Trigger_Falling EXTI_Trigger_Rising_Falling	**F:** Chọn cạnh tích cực Chọn cạnh lên Chọn cạnh xuống Chọn cạnh lên và xuống
A.EXTI_LineCmd = G; (Lệnh cho phép hoặc cấm kênh ngắt ngoài/sự kiện đang được cấu hình)	
G: ENABLE DISABLE	**G:** Cho phép hoặc cấm Cho phép Cấm
EXTI_Init(&A); (Lệnh cấu hình các thông số được lưu trong biến A cho EXTI)	

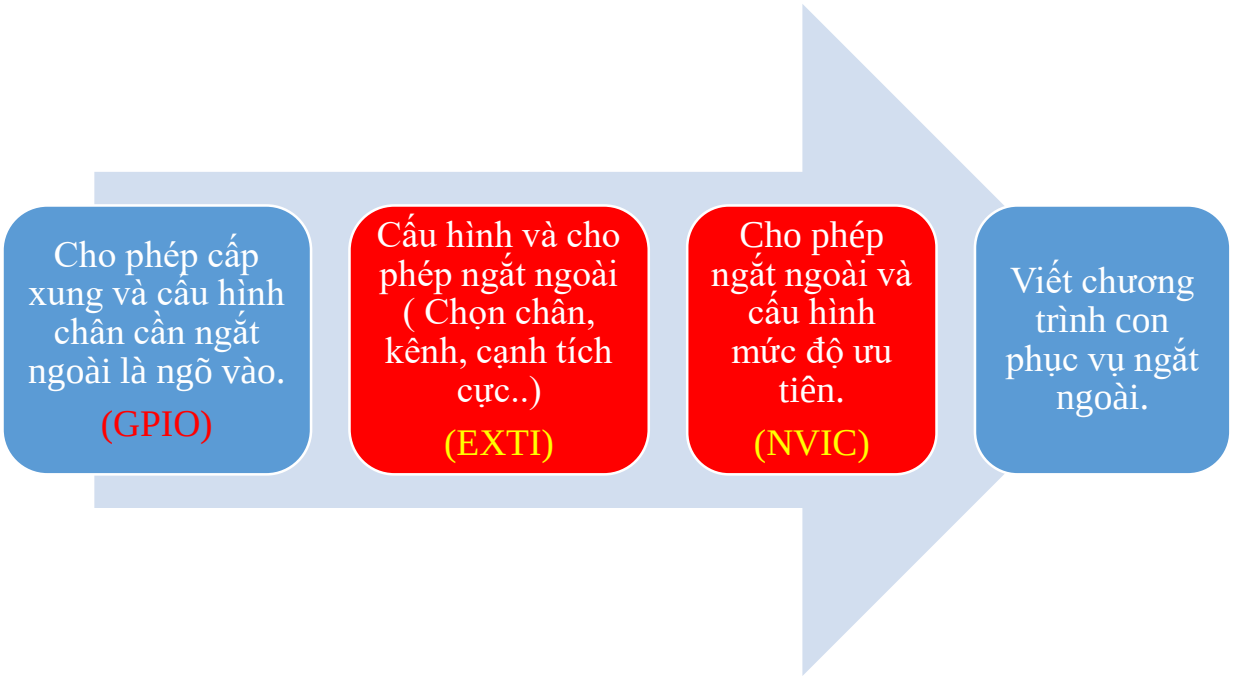
5.6.3 Ví dụ liên quan đến ngắt ngoài

Ví dụ 5.1: Viết chương trình cấu hình để cho phép chân **A8** nhận tín hiệu ngắt ngoài tác dụng **cạnh xuống**.

Chương trình

```
void CauhinhEXTI_A8()
{
  EXTI_InitTypeDef        EXTI_InitStructure;
  GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource8);
  // Chọn chân A8
  EXTI_InitStructure.EXTI_Line = EXTI_Line8;
  // Chọn kênh 8 - Do đang cấu hình cho chân A8
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  // Chọn chế độ ngắt ngoài
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  // Chọn tích cực cạnh xuống
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  // Cho phép ngắt ngoài
  EXTI_Init(&EXTI_InitStructure);
  // Cấu hình các cài đặt trên cho EXTI
}
```

5.6.4 Các bước để cấu hình và sử dụng ngắt ngoài cho 1 chân ARM STM32F1X



Hình 5.3 Các bước để cấu hình và sử dụng ngắt ngoài cho 1 chân ARM STM32F1X

5.7 CÁC VÍ DỤ LIÊN QUAN ĐẾN NGẮT VÀ SỰ KIỆN

5.7.1 Ví dụ sử dụng ngắt ngoài giao tiếp với nút nhấn

Ví dụ 5.2: Viết chương trình điều khiển LED kết nối với chân D9 sáng hoặc tắt nhờ hai nút nhấn ON và OFF được kiểm tra bằng ngắt ngoài. Biết nút ON được nối với chân C13 và nút OFF nối với chân A8.

Chương trình

```
#include "stm32f10x.h"
/* Chương trình con cấu hình chân D9 là ngõ ra
    đẩy kéo tốc độ 50 Mhz để điều khiển LED */
void cauhinhLED()
{
    GPIO_InitTypeDef      GPIO_InitStructure;
    RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOD,ENABLE );
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure .GPIO_Mode  = GPIO_Mode_Out_PP  ;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz ;
    GPIO_Init(GPIOD,&GPIO_InitStructure);
}

/* Cấu hình ngắt ngoài cho 2 chân A8 và C13
   Xem lại hình 5.3 thứ tự các bước cấu hình
     và sử dụng ngắt ngoài cho chân ARM ST*/

/* Bước 1 cấp xung và cấu hình chân A8, C13
 nhận tín hiệu ngắt ngoài là ngõ vào kéo lên
            (Cấu hình ở GPIO)              */
void cauhinhGPIOchoEXTI()
{
    GPIO_InitTypeDef    GPIO_InitStructure;
    RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA
                           |RCC_APB2Periph_GPIOC
                           |RCC_APB2Periph_AFIO,ENABLE);
    //Cho phép cấp xung clock cho PORTA, PORTC và AFIO
    //do ngắt ngoài là chức năng thay thế nên phải bật AFIO
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure .GPIO_Mode  = GPIO_Mode_IPU ;
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    // Cấu hình chân A8
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
    GPIO_Init(GPIOC,&GPIO_InitStructure);
    // Cấu hình chân C13
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource8);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,GPIO_PinSource13);
}
/* Bước 2 cấu hình và cho phép ngắt ngoài ở EXTI */
void cauhinhEXTI()
{
 EXTI_InitTypeDef      EXTI_InitStructure;
```

```
 EXTI_InitStructure.EXTI_Line = EXTI_Line8|EXTI_Line13;
 // Chọn kênh 8 và kênh 13 ứng với A8 và C13
 EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
 EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling ;
 EXTI_InitStructure.EXTI_LineCmd = ENABLE;
 EXTI_Init(&EXTI_InitStructure);
}
/* Bước 3 cấu hình cấp độ ưu tiên và cho phép
              ngắt ngoài ở NVIC                */
void cauhinhNVICchoEXTI()
{
    NVIC_InitTypeDef        NVIC_InitStructure;
    #ifdef  VECT_TAB_RAM
      NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
    #else
      NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
    #endif
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0 );
    NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn ;
    // Chọn các kênh từ 5-9
    NVIC_InitStructure.NVIC_IRQChannelSubPriority =0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    // Cấu hình NVIC cho các kênh 5-9 được phép ngắt
    NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn ;
    // Chọn các kênh từ 10-15
    NVIC_InitStructure.NVIC_IRQChannelSubPriority =1;
    NVIC_Init(&NVIC_InitStructure);
    // Cấu hình NVIC cho các kênh 10-15 được phép ngắt
}
/* Bước 4 viết chương trình con phục vụ ngắt ngoài*/
/* Chương trình con phục vụ ngắt cho chân A8*/
void EXTI9_5_IRQHandler()
{
  if(EXTI_GetITStatus(EXTI_IMR_MR8) != RESET)
  // Kiểm tra có phải là kênh 8 ngắt không?
    {  EXTI_ClearITPendingBit(EXTI_IMR_MR8);
         // Xóa cờ ngắt kênh 8
         GPIO_SetBits(GPIOD,GPIO_Pin_9);
         // Mở LED
    }
}
/*  Chương trình con phục vụ ngắt cho chân C13*/
void EXTI15_10_IRQHandler()
{
  if(EXTI_GetITStatus(EXTI_IMR_MR13) != RESET)
  // Kiểm tra có phải là kênh 13 ngắt không?
    {  EXTI_ClearITPendingBit(EXTI_IMR_MR13);
       // Xóa cờ ngắt kênh 13
         GPIO_ResetBits(GPIOD,GPIO_Pin_9);
```

```
            // Tắt LED
    }
}
int main(void)
{
    SystemInit ();
    cauhinhLED();
    cauhinhGPIOchoEXTI() ;
    cauhinhEXTI();
    cauhinhNVICchoEXTI();
    while(1);
}
```

Chú ý: Khi thực hiện lại ví trên ta cần thêm các thư viện sau vào Project: "**stm32f10x_rcc.c**" (các hàm cài đặt xung clock), "**misc.c**"(các hàm của NVIC), "**stm32f10x_gpio.c**"(các hàm điều khiển GPIO), "**stm32f10x_exti.c**"(các hàm điều khiển ngắt ngoài).

5.7.2 Ví dụ sử dụng sự kiện tác động từ bên ngoài để đánh thức CPU khỏi chế độ Stop

Ví dụ 5.3: Viết chương trình điều khiển CPU vào chế độ **Stop** sử dụng thư viện "stm32f10x_pwr"(xem bài 2). Khi có sự kiện tác động vào chân A8 thì đánh thức CPU và điều khiển LED đơn kết nối với chân D9 chớp tắt rồi lại tiếp tục cho CPU vào chế độ Stop.

Chương trình

```
#include<stm32f10x.h>
void delay(unsigned long t){while(t--);}
void cauhinhGPIO()
{
        GPIO_InitTypeDef  GPIO;
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA
            |RCC_APB2Periph_GPIOD|RCC_APB2Periph_AFIO,ENABLE );
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
        // Cho phép cấp nguồn cho bộ điều khiển năng lượng
        GPIO.GPIO_Pin = GPIO_Pin_8;
        GPIO.GPIO_Mode = GPIO_Mode_IPU ;
        GPIO_Init (GPIOA,&GPIO);
        GPIO.GPIO_Pin= GPIO_Pin_9;
        GPIO.GPIO_Mode = GPIO_Mode_Out_PP ;
        GPIO.GPIO_Speed = GPIO_Speed_50MHz ;
        GPIO_Init (GPIOD,&GPIO);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,
                                                GPIO_PinSource8 );
}
void cauhinhEXTI()
{
        EXTI_InitTypeDef   NN;
        NN.EXTI_Line= EXTI_Line8  ;
        NN.EXTI_Mode= EXTI_Mode_Event; // Chế độ sự kiện
        NN.EXTI_Trigger = EXTI_Trigger_Falling ;
        NN.EXTI_LineCmd = ENABLE;
```

```
        EXTI_Init(&NN);
}

int main()
{
        SystemInit();
        cauhinhGPIO();
        cauhinhEXTI();
        while(1)
        {
          // Đưa CPU vào chế độ Stop bằng WFE
             PWR_EnterSTOPMode(PWR_Regulator_LowPower
                                        ,PWR_STOPEntry_WFE);
          //Khi CPU thoát khỏi Stop thì cấu hình lại xung clock
             SystemInit();
             GPIO_SetBits(GPIOD,GPIO_Pin_9);delay(5000000);
             GPIO_ResetBits(GPIOD,GPIO_Pin_9); delay(5000000);
       }
}
```

Chú ý: Do khi vào chế độ Stop các bộ giao động PLL, HSI, HSE bị tắt hết nên khi thoát khỏi chế độ này ta nên cấu hình lại cho các bộ giao động trên. Các cấu hình cho ngoại vi do được lưu trong SRAM mà dữ liệu trong vùng nhớ này không mất khi Stop nên ta không cần cấu hình lại.

5.7.3 Ví dụ đưa CPU vào chế độ Standby và đánh thức CPU khỏi chế độ Standby

Ví dụ 5.4: Viết chương trình điều khiển CPU vào chế độ **Standby**(xem bài 2) bằng nút nhấn **SB** kết nối với chân A8 tác động bằng ngắt ngoài. Khi nhấn nút Wakeup thì đánh thức CPU và điều khiển LED đơn kết nối với chân D9 chớp tắt liên tục. Khi nhấn nút **SB** lần nữa thì CPU tiếp tục Standby.

Chương trình

```
#include "stm32f10x.h"
void delay(unsigned long t){while(t--);}
void cauhinhLED()
{    GPIO_InitTypeDef       GPIO_InitStructure;
     RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOD,ENABLE );
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
     GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_Out_PP  ;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz ;
     GPIO_Init(GPIOD,&GPIO_InitStructure);
}
void cauhinhGPIOchoEXTI()
{    GPIO_InitTypeDef    GPIO_InitStructure;
     RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA
                              |RCC_APB2Periph_AFIO,ENABLE);
     RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
     GPIO_InitStructure .GPIO_Mode  = GPIO_Mode_IPU ;
```

```
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource8);
}
void cauhinhEXTI()
{   EXTI_InitTypeDef      EXTI_InitStructure;
    EXTI_InitStructure.EXTI_Line = EXTI_Line8;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling ;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
}
void cauhinhNVICchoEXTI()
{   NVIC_InitTypeDef        NVIC_InitStructure;
    #ifdef  VECT_TAB_RAM
    NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
    #else
      NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
    #endif
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0 );
    NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority =0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
void EXTI9_5_IRQHandler()
{
  if(EXTI_GetITStatus(EXTI_IMR_MR8) != RESET)
    {    EXTI_ClearITPendingBit(EXTI_IMR_MR8);
         PWR_EnterSTANDBYMode();// Đưa CPU vào chế độ Standby
    }
}
int main(void)
{   SystemInit ();
    cauhinhLED();
    cauhinhGPIOchoEXTI() ;
    cauhinhEXTI();
    cauhinhNVICchoEXTI();
    PWR_WakeUpPinCmd(ENABLE );
    // Cho phép nút chân Wakeup đánh thức CPU
    while(1)
    {
        GPIO_SetBits(GPIOD,GPIO_Pin_9);   delay(5000000);
        GPIO_ResetBits(GPIOD,GPIO_Pin_9); delay(5000000);
    }
}
```

Chú ý: Khi vào chế độ Standby các bộ giao động PLL, HSI, HSE đều bị tắt, dữ liệu trong SRAM cũng bị xóa nên khi thoát khỏi chế độ này ta nên cấu hình lại cho các bộ giao động và các ngoại vi cần sử dụng. Trong **ví dụ 5.4** do sau khi thoát khỏi chế độ Standby chương trình sẽ được thực thi lại từ đầu chương trình chính nên các chương trình con cấu hình được gọi lại.