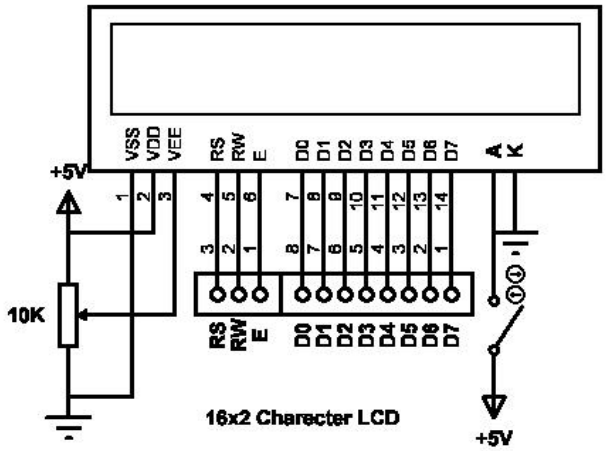
9 آشنایی با LCD کارکتری 2*16

LCD 16x2 یک نمایشگر کریستالی مایع است که معمولاً در پروژه‌های الکترونیکی و میکروکنترلرها مورد استفاده قرار می‌گیرد. این نمایشگر قادر است 16 کاراکتر را در هر خط و 2 خط را نمایش دهد. ویژگی‌های اصلی این نمایشگر عبارتند از:

1. ابعاد : 2x16 به این معنی است که نمایشگر 16 کاراکتر در هر خط و 2 خط دارد.
2. نوع کاراکتر : معمولاً از کاراکترهای ASCII و کاراکترهای خاص برای نمایش استفاده می‌شود.
3. تغذیه : این نمایشگر معمولاً با ولتاژ 5 ولت کار می‌کند و نیاز به یک منبع تغذیه مناسب دارد.
4. پروتکل ارتباطی : ارتباط با میکروکنترلرها معمولاً از طریق پروتکل‌های سریال یا موازی انجام می‌شود.
5. کاربردها : در دستگاه‌های مختلفی مانند دستگاه‌های اندازه‌گیری، کنترلرهای صنعتی استفاده می‌شود.
6. نحوه کار : برای نمایش اطلاعات بر روی LCD، باید داده‌ها به صورت باینری به آن ارسال شوند و سپس می‌توان از دستورات خاصی برای کنترل نمایشگر استفاده کرد.

9.1 پایه‌ها و وظایف آن‌ها:

LCD 16x2 دارای 16 پایه است که هر کدام وظیفه خاصی دارند. در زیر به توضیح پایه‌های این نمایشگر می‌پردازیم:



1. پایه 1 (VSS) :

- زمین (GND) نمایشگر است و باید به منبع تغذیه منفی متصل شود.

2. پایه 2 (VDD) :

- ولتاژ مثبت (معمولاً 5 ولت) که برای تغذیه نمایشگر استفاده می‌شود.

3. پایه 3 (VO) :

- این پایه برای تنظیم کنتراست نمایشگر استفاده می‌شود. با اتصال یک پتانسیومتر به این پایه، می‌توان کنتراست نمایش را تنظیم کرد.

4. پایه 4 (RS) :

- این پایه انتخاب رجیستر است. اگر این پایه HIGH باشد، داده‌های ارسال شده به عنوان داده کاراکتری در نظر گرفته می‌شوند و اگر LOW باشد، داده‌ها به عنوان دستورالعمل (Command) در نظر گرفته می‌شوند.

5. پایه 5 (RW) :

- این پایه برای تعیین حالت خواندن یا نوشتن است. اگر LOW باشد، حالت نوشتن فعال است و اگر HIGH باشد، حالت خواندن فعال است.

6. پایه 6 (E) :

- این پایه برای فعال‌سازی است. با ارسال یک پالس به این پایه، داده‌ها یا دستورات به نمایشگر ارسال می‌شوند.

7. پایه‌های 7 تا 14 (D0-D7) :

- این پایه‌ها برای ارسال داده‌های باینری به نمایشگر استفاده می‌شوند. در حالت 8 بیتی، از پایه‌های D0 تا D7 استفاده می‌شود و در حالت 4 بیتی، فقط از D4 تا D7 استفاده می‌شود.

8. پایه 15 (A) :

- این پایه برای تأمین نور پس‌زمینه (Backlight) نمایشگر است و باید به ولتاژ مثبت متصل شود.

9. پایه 16 (K) :

- این پایه زمین برای نور پس‌زمینه است و باید به زمین (GND) متصل شود.

9.2 فایل کتابخانه‌ای LCD

فایل کتابخانه‌ای LCD شامل تعدادی زیربرنامه است که کارکردن با LCD را ساده تر می نماید. فایل STM_MY_LCD16X2.h و STM_MY_LCD16X2.c در سامانه یکتا بارگذاری شده است.

نکته: در این فایل کتابخانه پایه RW از LCD به زمین وصل شده است تا تعداد پایه های مورد نیاز در حالت داده 4 بیتی و 8 بیتی کاهش یابد.

تعدادی از زیربرنامه ها به شرح ذیل است.

9.2.1 توابع خصوصی

1 : LCD1602_EnablePulse() .یک پالس بر روی پین E تولید می‌کند تا داده‌ها به LCD منتقل شوند.
2 : LCD1602_RS() .وضعیت پین RS را تنظیم می‌کند تا مشخص کند داده ارسالی فرمان است یا داده.
3 : LCD1602_write() .یک بایت داده را به LCD ارسال می‌کند، چه در حالت ۸ بیت و چه در حالت ۴ بیت.
4 : LCD1602_TIM_Config() .تایمر را برای تولید تأخیر در میکروثانیه‌ها پیکربندی می‌کند.
5 : LCD1602_TIM_MicorSecDelay() .تأخیری به مدت مشخصی در میکروثانیه‌ها ایجاد می‌کند.
6 : LCD1602_writeCommand() .یک فرمان به LCD ارسال می‌کند با تنظیم RS به ۰ و فراخوانی تابع نوشتن.
7 : LCD1602_writeData() .داده‌ای به LCD ارسال می‌کند با تنظیم RS به ۱ و فراخوانی تابع نوشتن.
8 : LCD1602_write4bitCommand() .یک فرمان را در حالت ۴ بیت ارسال می‌کند.

9.2.2 توابع عمومی

1. LCD1602_Begin8BIT() :این تابع نمایشگر را در حالت ۸ بیت راه‌اندازی می‌کند و دستورات اولیه لازم را ارسال می‌کند.

2. LCD1602_Begin4BIT()این تابع نمایشگر را در حالت ۴ بیت راه‌اندازی می‌کند.
3. LCD1602_print() :یک رشته را به LCD چاپ می‌کند، به صورت کاراکتر به کاراکتر.
4. LCD1602_setCursor() :مکان‌نما را به ردیف و ستونی مشخص منتقل می‌کند.
5. LCD1602_1stLine() / LCD1602_2ndLine() :مکان‌نما را به ابتدای خط اول یا دوم منتقل می‌کند.
6. LCD1602_TwoLines() / LCD1602_OneLine() :نمایش را برای نشان دادن یک یا دو خط تنظیم می‌کند.
7. LCD1602_noCursor() / LCD1602_cursor() :مکان‌نما را فعال یا غیرفعال می‌کند.

8. LCD1602_clear() : نمایش را پاک می‌کند.
9. LCD1602_shiftToRight() / LCD1602_shiftToLeft() : محتویات نمایش را به سمت چپ یا راست جابجا می‌کند.
10. LCD1602_PrintInt() / LCD1602_PrintFloat() : توابعی برای چاپ مقادیر صحیح و اعشاری بر روی LCD.

9.3 پیکربندی میکروبرای اتصال به LCD

در هر پروژه بخش RCC و کلاک میکرو را تنظیم نمایید.

ابتدا تنظیمات پین‌ها را مطابق ذیل انجام دهید.

پین‌های مربوط به RS، RW، E، D0-D7 را در STM32CubeMX تنظیم کنید. معمولاً از پین‌های GPIO برای این کار استفاده می‌شود. پین‌ها را به صورت زیر تنظیم کنید:

RS: خروجی

RW: خروجی

E: خروجی

D0-D7: خروجی (در حالت 4 بیتی فقط D4-D7 را استفاده کنید)

در بدنه اصلی برنامه نیز این پایه ها در تعریف برنامه پیکربندی LCD با عنوان LCD1602_Begin4BIT در آرگومان های ورودی مانند برنامه 9-1 به درستی تنظیم گردد.

الف	void LCD1602_Begin4BIT(GPIO_TypeDef* PORT_RS_E, uint16_t RS, uint16_t E, GPIO_TypeDef* PORT_MSBs4to7, uint16_t D4, uint16_t D5, uint16_t D6, uint16_t D7)
ب	LCD1602_Begin4BIT(GPIOA, GPIO_PIN_0, GPIO_PIN_1, GPIOA, GPIO_PIN_3, GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6);
ج	Rs: PA0 E: PA1 RW: GND D4: PA3 D5: PA4 D6: PA5 D7: PA6

	برنامه 9-1: الف: ساختار زیربرنامه پیکربندی LCD،ب: مثال از نحوه استفاده از زیربرنامه، ج: اتصالات میکرو و LCD بر اساس تنظیمات زیربرنامه

9.4 پروژه نمایش داده روی LCD

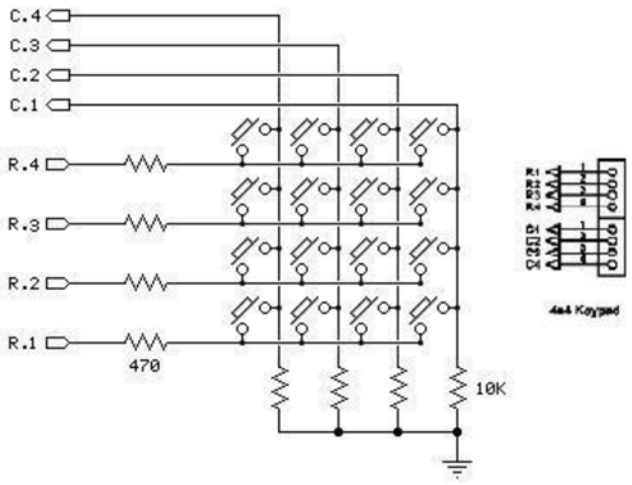
در ادامه میتوان از سایر زیربرنامه های معرفی شده در کتابخانه برای نمایش روی LCD مانندبرنامه 9-2 استفاده نمود. فایل STM_MY_LCD16X2.h روی سامانه یکتا قرار دارد. این هدر فایل را دانلود و در پوشه inc در محل تعریف پروژه ذخیره نمایید. فایل با فرمت C را هم در پوشه src کپی نمایید.

```
#include "main.h"
#include "STM_MY_LCD16X2.h"

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  LCD1602_Begin4BIT(GPIOA, GPIO_PIN_0, GPIO_PIN_1, GPIOA, GPIO_PIN_3, GPIO_PIN_4,
GPIO_PIN_5, GPIO_PIN_6);

  while (1)
  {
          LCD1602_clear();
         LCD1602_print("Microlab ");
          HAL_Delay(500);
          LCD1602_2ndLine();
          LCD1602_print("ECE.IUT.AC.IR");

          HAL_Delay(500);
          LCD1602_clear();
          LCD1602_PrintInt(12345);
          HAL_Delay(500);
          LCD1602_2ndLine();
          LCD1602_PrintFloat(14.567894,6);
          HAL_Delay(500);
  }
 }
```

برنامه 9-2: نمایش داده روی LCD

9.5 آشنایی با صفحه‌کلید ماتریسی

صفحه کلید یکی از متداول‌ترین ادوات ورودی برای دریافت داده‌ها از سوی کاربر می‌باشد. در پکیج آموزشی یک عدد صفحه‌کلید ماتریسی شامل 16 عدد کلید فشاری که چیدمان آن‌ها به صورت 4x4 (4 ردیف و 4 ستون) می‌باشد قرار داده شده است. شماتیک مربوط به مدار keypad در شکل 9-1 مشاهده می‌شود.



شکل 9-1: مدار صفحه کلید ماتریسی استفاده شده در بورد آموزشی

برای کار با keypad پایه‌های آن به یکی از درگاه‌های ریزپردازنده متصل می‌شود به این صورت که سطرهای R1 تا R4 به صورت خروجی و ستون‌های C1 تا C4 به صورت ورودی تعریف می‌گردند. سپس مقدار متناظر هر کلید به صورت یک آرایه در ریزپردازنده ذخیره می‌گردد. به عنوان مثال، در برنامه 9-3برنامه 9-3، ماتریس data_key به ازای هر یک از کلیدهای keypad یک مقدار متناظر را در برگرفته است.

برنامه 9-3
```
char data_key[]={
'7','8','9','a',
'4','5','6','b',
'1','2','3','c',
'*','0','#','d'};
```

شناسایی کلید فشرده شده در سه گام صورت میگیرد:

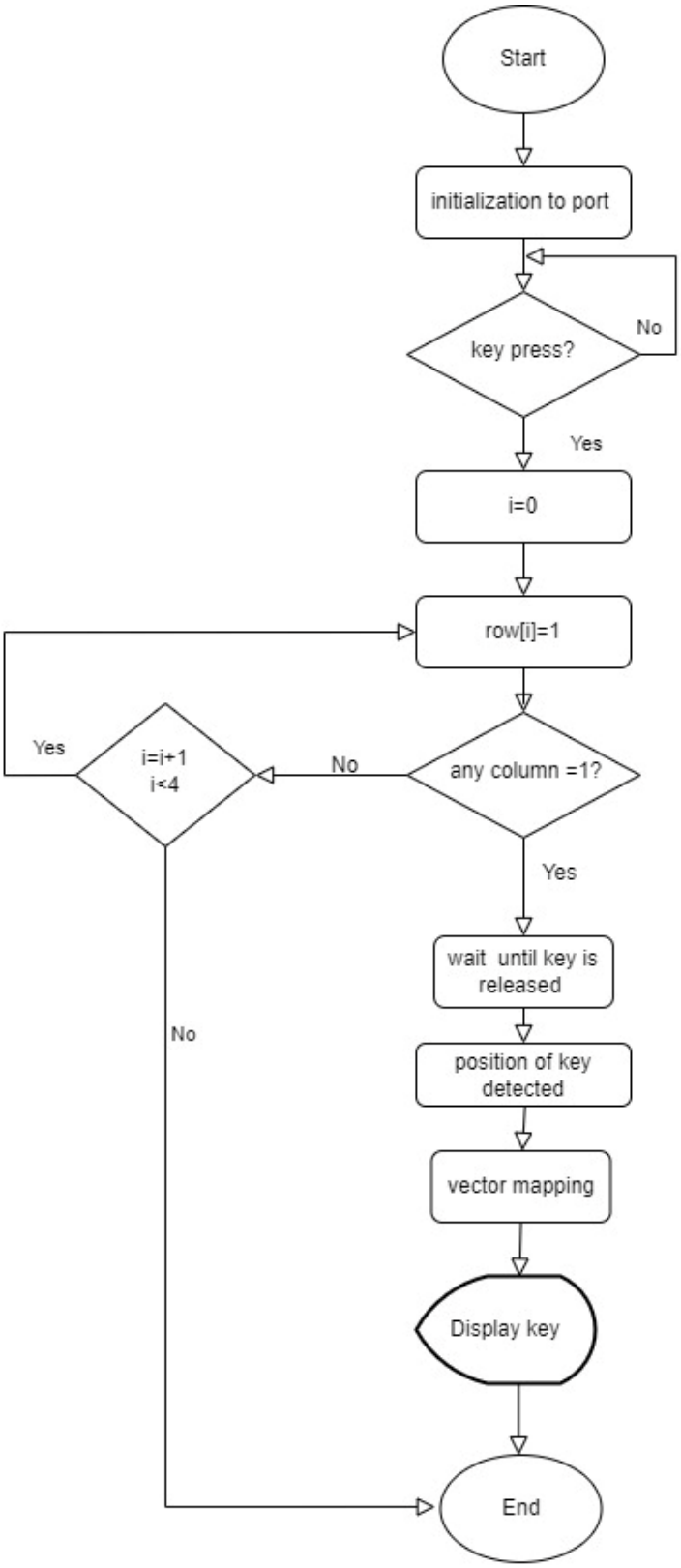
1- شناسایی فشردن کلید
2- تعیین جایگاه کلید فشرده شده

3- نگاشت یک کارکتر به موقعیت کلید

برای مرحله اول کافی است که مقدار اولیه ای روی پورت مورد نظر قرار گیرد و تا زمانی که این مقدار پا برجا باشد کلیدی فشرده نشده است.

پس از شناسایی فشرده شدن کلید، پردازنده وارد گام بعدی که تعیین جایگاه کلید است ، میگردد . بدین منظور مانند شکل 9-2 سطرها به ترتیب 1 گردیده و مقدار ستون‌ها خوانده می‌شود. هر جا که مقدار ستونی 1 شده باشد، یعنی کلید مربوط به آن سطر و ستون فشرده شده است. فرض کنید کلید 6 فشرده شده باشد، در این صورت با یک شدن پایه نظیر سطر دوم، مقدار پایه متناظر با ستون سوم نیز مقدار 1 و سایر ستون‌ها مقدار 0 را برمی‌گردانند. با پیدا شدن سطر و ستون، می‌توان از طریق ماتریس ذخیره شده در ریزپردازنده، به محتوای کلید دست پیدا کرد.

```
data_key[(2 - 1) * 4 + (3 - 1)] = '6'
```



شکل 9-2: روند نمای شناسایی کلید فشرده شده در کیپد 4*4

این روش کار با صفحه کلید را روش سرکشی می‌نامند که یک نمونه کد برای تعیین کلید فشرده شده به این طریق در برنامه 4-9 نشان داده شده است. در این برنامه keypad_write زیربرنامه نوشتن در پورت متصل به کیپد هست که با توجه به درگاه مورد استفاده و یا استفاده از دستورات HAL و یا نوشتن در سطح رجیستری میتواند متفاوت باشد. keypad_read نیز خواندن درگاه مورد نظر است. Keypad جایگاه کلید فشرده شده را شناسایی میکند و LCD1602_print نیز کلید فشرده شده را روی lcd نشان میدهد.

```
#include "main.h"
#include "STM_MY_LCD16X2.h"
#include "keypad.h"
#include "stdio.h"

#define PORTKey         GPIOC

char data_key[16]={
'7','8','9','/',
'4','5','6','*',
'1','2','3','+',
'C','0','=','-'};

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
  char str[16];

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

LCD1602_Begin4BIT(GPIOE, GPIO_PIN_0, GPIO_PIN_1, GPIOE, GPIO_PIN_3, GPIO_PIN_4,
GPIO_PIN_5, GPIO_PIN_6);
 LCD1602_clear();
 LCD1602_print("Microlab ");
HAL_Delay(500);
while (1)
  {
                    keypad_write(0xf0);
                    while(keypad_read()==0xf0);
                    int keypress=keypad();
                    if (keypress!=100)
                    {
                    sprintf(str,"%c",data_key[keypress]);
                    LCD1602_print(str);
    }
    }
}
```

```
char keypad(void)
{
  char row[4]={0x10,0x20,0x40,0x80};
  char key=100;
  char c;

   for (r=0;r<4;r++)
   {
    keypad_write(row[r]);

   c=20;

   if (keypad_read_pin(0)==1) c=0;
   if (keypad_read_pin(1)==1) c=1;
   if (keypad_read_pin(2)==1) c=2;
   if (keypad_read_pin(3)==1) c=3;


      if (!(c==20)){
         key=(r*4)+c;
         keypad_write(0xf0);
         while (keypad_read_pin(0)==1) {}
         while (keypad_read_pin(1)==1) {}
         while (keypad_read_pin(2)==1) {}
         while (keypad_read_pin(3)==1) {}
      }
    keypad_write(0xf0);
   }
   return key;
}
```

برنامه 9-4: برنامه نمایش کلید فشرده شده در کیپد 4*4

در زیربرنامه keypad نیز ، keypad_read_pin بیت خاصی از پورت را میخواند. نحوه کدنویسی هر یک از این زیربرنامه ها به کاربر بستگی دارد به عنوان نمونه تعدادی از زیربرنامه ها را میتوان به صورت ذیل تعریف کرد.

```
void keypad_write(int data)
{
                PORTKey->ODR =(PORTKey->ODR & 0xffffff00) | (data & 0x000000ff);
}
```

```
int keypad_read_pin( int pin)
{
        return (PORTKey->IDR >> pin) & 0x00000001;
}
```

```
int keypad_read( void)
{
  return (PORTKey->IDR & 0x000000FF);
}
```

برای نمایش روی LCD و پیکربندی LCD نیز از توابع موجود در STM_MY_LCD16X2 استفاده نمایید.

9.6 پروژه طراحی ماشین حساب

یک ماشین حساب ساده با چهار عمل اصلی برای اعداد تک رقمی طراحی نمایید و نتیجه عملیات را روی LCD نشان دهید. از نگاشت ذیل استفاده نمایید.

```
char data_key[]={
'7','8','9','+',
'4','5','6','-',
'1','2','3','×',
'C','0','=','/'};
```

9.7 وقفه های خارجی

وقفه‌های خارجی (External Interrupts) در میکروکنترلر STM32F407 یکی از قابلیت‌های کلیدی هستند که به برنامه‌نویسان اجازه می‌دهند تا به رویدادهای خارجی به سرعت واکنش نشان دهند. در زیر به توضیح این وقفه‌ها و نحوه کار آنها می‌پردازیم:

9.7.1 تعریف وقفه‌های خارجی

وقته‌های خارجی به سیگنال‌هایی اطلاق می‌شود که از منابع خارجی (مانند دکمه‌ها، سنسورها و ...) به میکروکنترلر ارسال می‌شوند. این وقفه‌ها می‌توانند به صورت لبه‌ای (rising edge یا falling edge) یا سطحی (high یا low) فعال شوند.

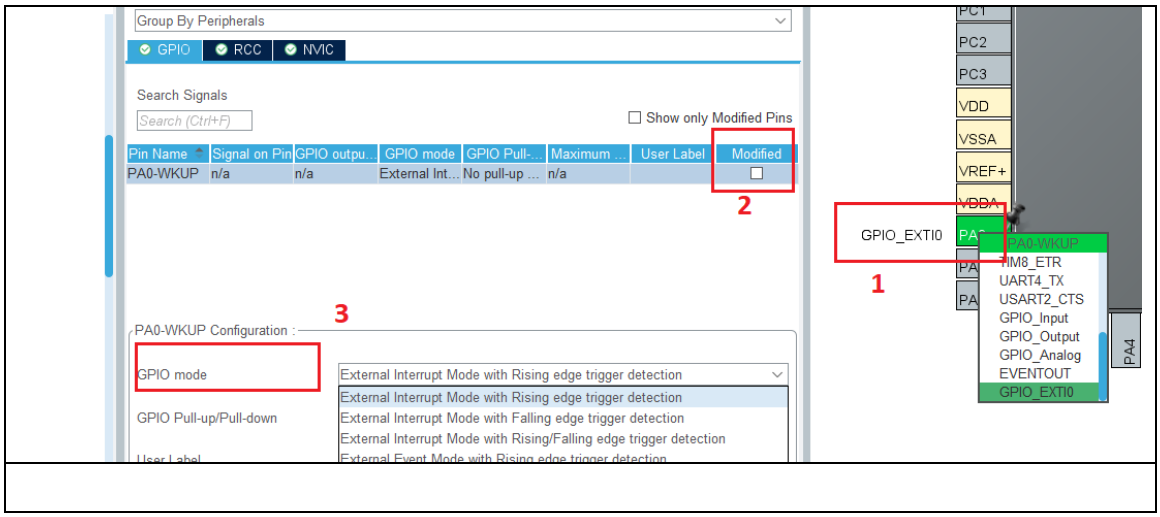
9.7.2 پیکربندی وقفه‌های خارجی

برای استفاده از وقفه‌های خارجی در STM32F407، مراحل زیر باید انجام شود:

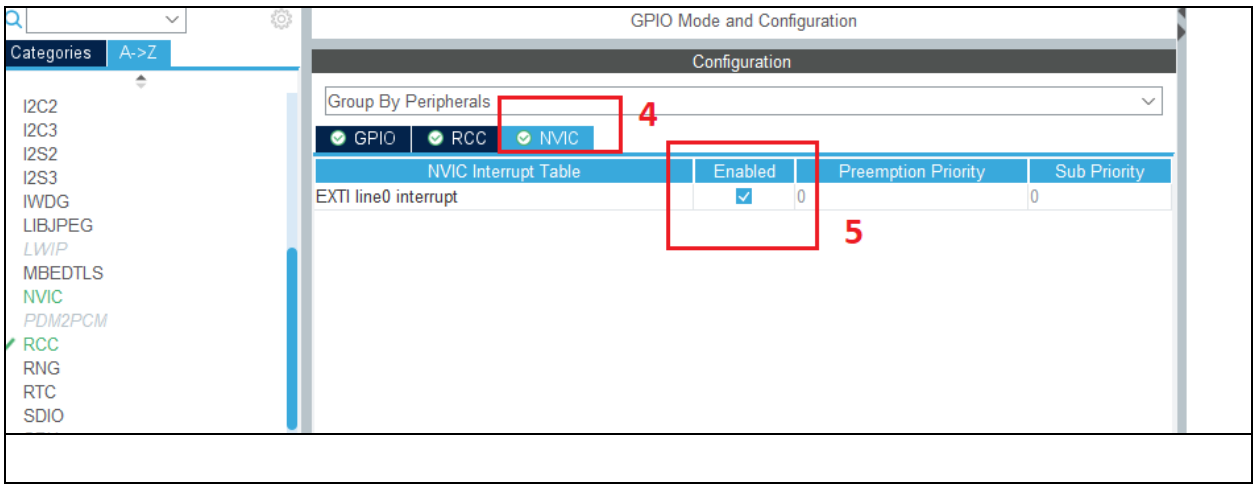
- فعال‌سازی کلاک : ابتدا باید کلاک مربوط به GPIO و SYSCFG فعال شود. بدین منظور میتوان از STM32cubeMx بهره گرفت.

- پیکربندی پین GPIO : پین GPIO که به عنوان ورودی وقفه استفاده می‌شود، باید به درستی پیکربندی شود. این شامل تنظیم حالت پین به ورودی و فعال‌سازی Pull-up یا Pull-down در صورت نیاز است. بدین منظور، در بخش "Pinout & Configuration"، پین مورد نظر برای وقفه خارجی (مثلاً `PA0`) را انتخاب کنید و آن را به عنوان "External Interrupt" تنظیم کنید.

- تنظیم نوع وقفه : نوع وقفه (لبه بالا، لبه پایین، یا سطح) باید مشخص شود.



- تنظیم اولویت وقفه : STM32F407 به شما اجازه می‌دهد تا اولویت‌های مختلفی برای وقفه‌ها تنظیم کنید. بدین منظور به بخش "Configuration" بروید و "NVIC" را انتخاب کنید. در اینجا می‌توانید اولویت وقفه را تنظیم کنید. مطمئن شوید که وقفه‌های مورد نیاز فعال شده‌اند.



9.7.3 مدیریت وقفه‌ها

پس از پیکربندی و ایجاد کد ها در محیط Keil مشاهده می‌شود یک تابع handler برای وقفه تعریف شده است. این تابع به محض وقوع وقفه، به طور خودکار فراخوانی می‌شود. در این تابع، می‌توانید اقدامات لازم را انجام دهید.

```
void EXTI0_IRQHandler(void) {
   if (EXTI->PR & (1 << 0)) { // بررسی وقوع وقفه
      // کد برای انجام کاری در هنگام فشار دادن دکمه
      EXTI->PR |= (1 << 0); // پاک کردن پرچم وقفه
   }
}
```

نکات مهم

- تداخل وقفه‌ها : در صورت وجود چندین وقفه، باید به ترتیب اولویت آنها توجه شود.
- مدیریت زمان : در داخل تابع handler، زمان زیادی صرف نکنید. بهتر است که یک پرچم تنظیم کنید و در حلقه اصلی به آن پاسخ دهید.

9.7.4 پروژه led چشمک زن با رخداد وقفه با توابع HAL

در این پروژه بسیار ساده یکی از کلیدها را به پایه PA0 متصل نموده و با تغییر وضعیت کلی LED متصل به پایه تغییر وضعیت میدهد.هدف از انجام این پروژه صرفا آشنایی با مدیریت وقفه ها می‌باشد.

```
#include "main.h"


void SystemClock_Config(void);
static void MX_GPIO_Init(void);

void GPIO_Init(void){

		RCC->AHB1ENR |= (1<<0);  // Enable the GPIOA clock
		GPIOA->MODER |= (1<<10);  // pin PA5(bits 11:10) as Output(01)
```


نسخه پیشنویس(ارسال اشکال فنی و نگارشی به zmohammadzadeh@of.iut.ac.ir)

```
                    //Configure the output mode i.e Output type, speed, and pull
                    GPIOA->OTYPER &= ~(1<<5);  // bit 5=0 --> Output push pull
                    GPIOA->OSPEEDR |= (1<<11);  // Pin PA5 (bits 11:10) as Fast Speed(1:0)
                    GPIOA->PUPDR &= ~((1<<10) | (1<<11));  // Pin PA5 (bits 11:10) are 0:0 --> no pull
up or pulldown
}

int main(void)
{

  HAL_Init();

  SystemClock_Config():
  MX_GPIO_Init(); //PA0 : external interrupt
  GPIO_Init();  // PA5 : output


HAL_NVIC_SetPriority(EXTI0_IRQn, 2,0);
HAL_NVIC_EnableIRQ(EXTI0_IRQn);


  while(1)
}
{

}
```

```
void EXTI0_IRQHandler(void) // this subroutine is in stm32fxx_it.c
{
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
          led_toggle();
{
```

```
void led_toggle(void)
{
GPIOA->BSRR |= (1<<5); // Set the Pin PA5

for(int i=0; i<50;i++)delay(1000000);
GPIOA->BSRR |= (1<<5) <<16;  // Clear the Pin PA5

for(int i=0; i<50;i++)delay(1000000);
}
```

9.8 پروژه led چشمک زن با رخداد وقفه با توابع LL

```
#include "main.h"

void EXTI0_IRQHandler(void) {
    if (EXTI->PR & (1 << 0)) {
       led_toggle();
        EXTI->PR |= (1 << 0);
                    }
           }

int main(void)
{
 SysClockConfig();
 GPIO_Init();

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
    GPIOA->MODER &= ~(3 << (0 * 2));

    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
    EXTI->IMR |= (1 << 0);
    EXTI->FTSR |= (1 << 0);



    NVIC_SetPriority(EXTI0_IRQn, 2);
    NVIC_EnableIRQ(EXTI0_IRQn);


while(1){
}
}
```