گزارش آزمایشگاه

رشته مهندسی برق

گزارشکار سری ششم آزمایشگاه ریزپردازنده و مدارهای واسطه

نگارش

امیرحسین منصوری

مهشاد اکبری سریزدی

آنوشا شریعتی

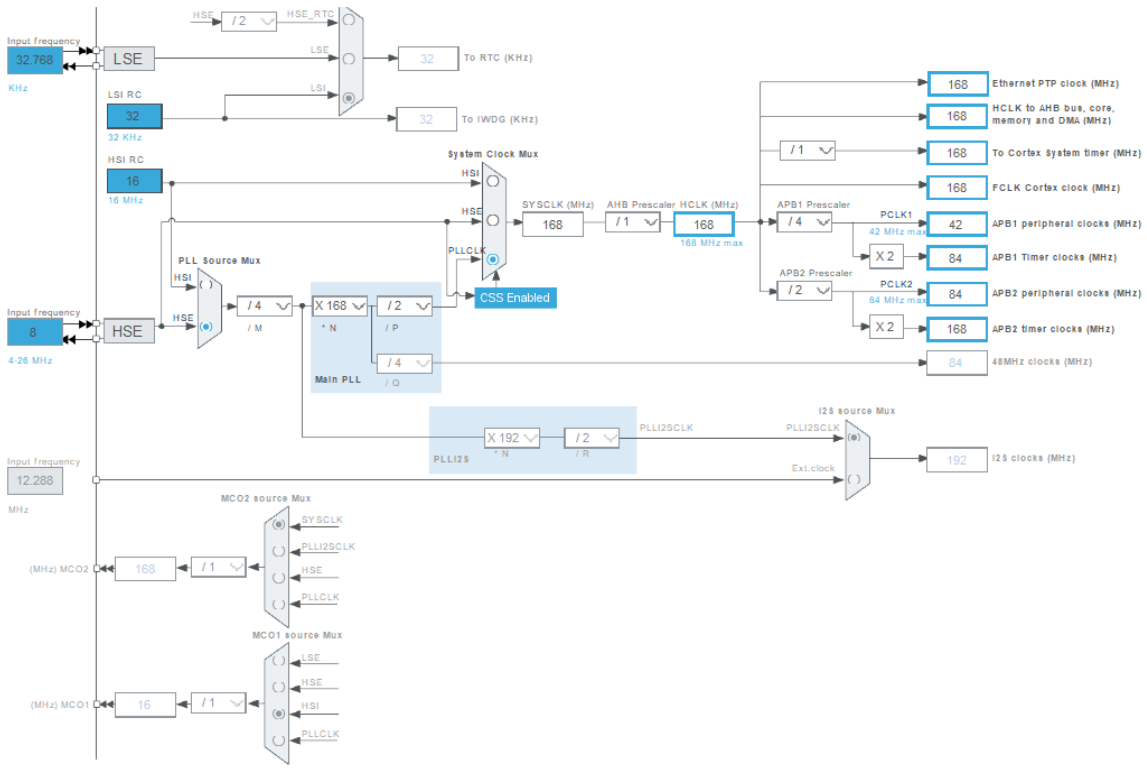
استاد راهنما

مهندس ذکی‌زاده

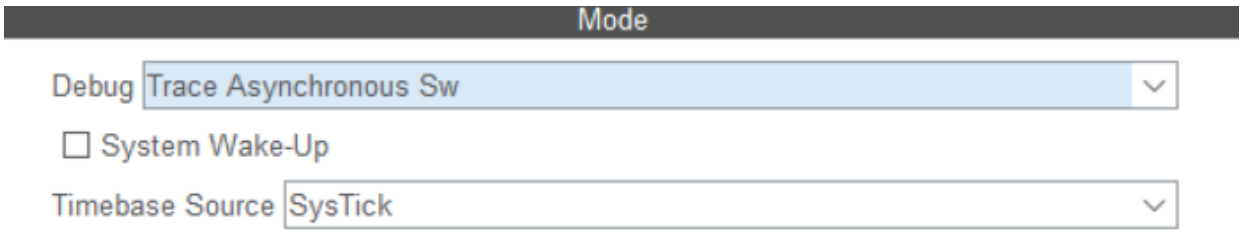
آذر ۱۴۰۲ شمسی

برای شروع کار مدل پردازنده *(STM32F407VGT6)* از منوی *MCU* انتخاب می‌کنیم.

سپس برای اینکه بتوانیم از میکرو استفاده کنیم و از مزایای *Debugging* آن بهره‌مند شویم باید تغییراتی را در کلاک دستگاه اعمال کنیم. برای تنظیم *HSE* دستگاه از یک نوسان‌ساز خارجی با فرکانس‌کاری *۸ MHz* استفاده می‌کنیم، سپس تامین کلاک پردازنده را از *PLL* تعیین کرده و منبع آنرا *HSE* قرار می‌دهیم. حداکثر فرکانس‌کاری این پردازنده *۱۶۸ MHz* می‌باشد. با قراردادن فرکانس‌کاری سایر تغییرات به صورت اتوماتیک انجام خواهد شد.



پس از این برای بهره مندی از امکانات دیباگینگ، مد دیباگ را به *Trace Asynchronous Sw* تغییر می‌دهیم.



حال که اعمال مقدماتی را انجام دادیم، کافی است گزینه *Generate-code* را فشار دهیم تا سایر تنظیمات توسط نرم‌افزار انجام شود.

بخش دوم گزارشکار پنجم:

در این بخش هدف ساخت یک تایمر با استفاده از شمارش تعداد کلاک‌های دستگاه است، برای این منظور از کدی مطابق زیر استفاده می‌کنیم و در ادامه به تشریح آن خواهیم پرداخت:

```
.global delay
delay:
 LDR R1, =#33600000 ;
 MUL R1, R0 ;
 B PROCESS ;
PROCESS:
 SUB R1, #1 ;
 CMP R1, #0 ;
 CBZ R1, RET ;
 B PROCESS
RET:
BX LR ;
```

این ساختار از ۴ دستور مکررا استفاده می‌کند که برای همین درنگاه اول به نظر می‌رسد با شمارس فرکانس کلاک پردازنده تقسیم بر ۴ میتوان به عملکرد مطلوب رسید، اما مشاهدات‌ما نشان می‌دهد در شمارش ۳ ثانیه در صورتی که تعداد کلاک‌ها را ۴ در نظر بگیریم منجر به خطا در شمارش خواهد شد، در صورتی که در نظرگرفتن ۵ کلاک خطا را به حداقل می‌رساند، با توجه به نکته ذکرشده میتوان نتیجه‌گیری کرد دستورات فوق ۵ کلاک را شامل میشوند که به احتمال زیاد به دلیل وجود دستور مقایسه است.

در ادامه قطعه‌کد دیگری برای ساخت که برای ساخت چراغ چشمک‌زن نوشته‌شده است، قرار می‌گیرد:

```
extern void led_on(void);
extern void led_off(void);
extern void delay(uint32_t t);
```

در ابتدا توابعی که به زبان اسمبلی نوشته‌ایم را به تابع *main* اضافه می‌کنیم. قطعه کد زیر در نهایت *LED* را به مدت ۳ ثانیه خاموش و روشن خواهد کرد.

```
 while (1)
 {
   /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
     led_on();
     delay(3);
     led_off();
     delay(3);

 }
 /* USER CODE END 3 */
```

گزارشکار ششم:

در این گزارشکار هدف این است که کارهایی که تاکنون انجام دادیم، به صورت بسیار آسان‌تر و توسط کتابخانه‌های *HAL* انجام دهیم.

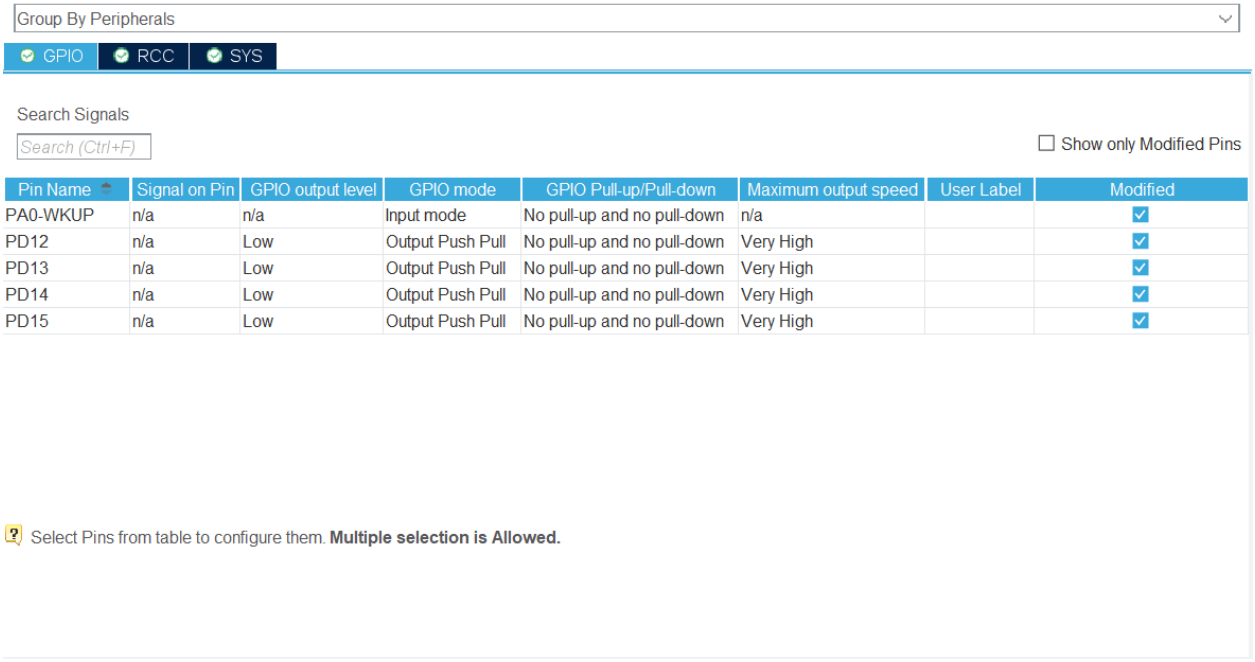
تمام‌کارهایی که توسط تغییر در مقادیر رجیسترها انجام دادیم، توسط توابعی به زبان *C* و به راحتی *initialize* می‌شوند، برای مثال تابع *MX_GPIO_Init* که در فایل *gpio.c* نوشته است و شامل *entity* های زیر است:

```
void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,

  /*Configure GPIO pin : PA0 */
  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PD12 PD13 PD14 PD15 */
  GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
}
```

- سرعت پایه‌ها
- وضعیت کاری پایه‌ها
- وضعیت *Pullup* و *Pulldown* پایه‌ها
- وضعیت اولیه پایه‌ها

بخش اول:

در این بخش ابتدا در *Cube MX* پایه‌های متصل به *LED*ها را در وضعیت *Output* و پایه متصل به *User* را در وضعیت *Input* قرار می‌دهیم:



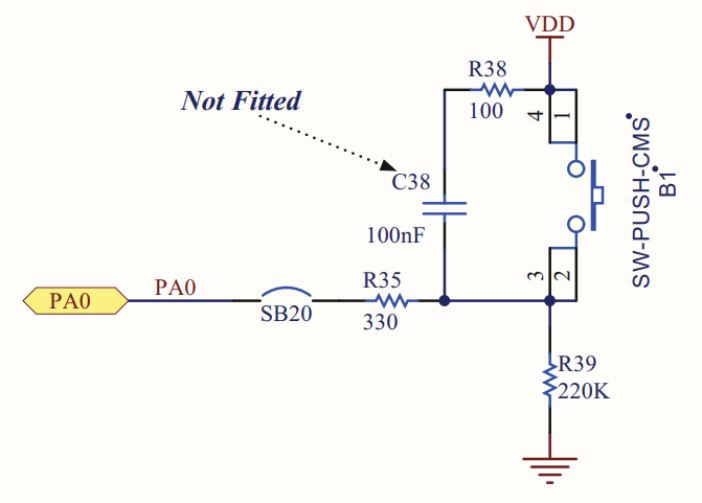
در ادامه قطعه کد زیر را خواهیم نوشت:

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    HAL_Delay (3000);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_12,1);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13,1);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,1);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,1);
    HAL_Delay (3000);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_12,0);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13,0);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,0);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,0);
}
/* USER CODE END 3 */
```

که *LED* ها را به صورت متناوب خاموش و روشن می‌کند.

بخش دوم:

شمای اتصالات خارجی دکمه *User* مطابق زیر است:



همانگونه که مشاهده می‌شود مدار فوق از نوع *Pull Down* است. در صورت *Pull Up* بودن هنگام فشردن دکمه ولتاژ صفر و هنگام آزاد بودن دکمه ولتاژ ۳.۳ ولت خواهد بود.

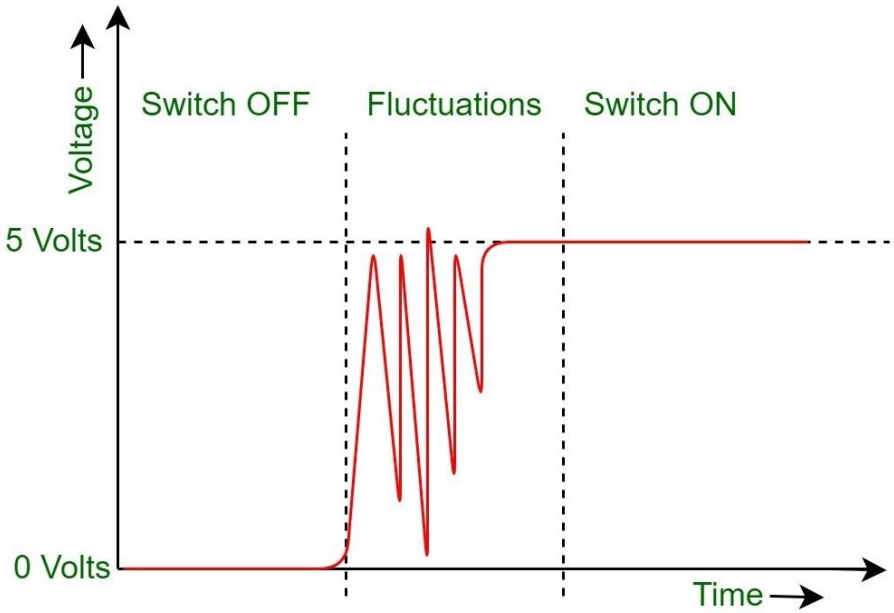
بخش سوم:

برای این بخش با توجه به *Pull Down* بودن دکمه مذکور شرطی قرار میدهیم تا در صورت صفر بودن پایه ورودی، چراغها خاموش و در غیر این صورت چراغها روشن باشند. شمای کد این بخش مطابق زیر است:

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

    if (HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)) {

        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_12,1);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13,1);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,1);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,1);
    }
    else{
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_12,0);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13,0);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,0);
        HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,0);

    }
```

بخش چهارم:

در این بخش با پدیده‌ای تحت عنوان پدیده کلیدزنی روبه‌رو هستیم، وضعیت یک کلید، مدت کوتاهی پس از فشرده شدن مطابق زیر است:



برای حل مشکل فوق می‌توان از روش‌های نرم افزاری و سخت‌افزاری استفاده کرد. یکی از روش‌های حال نرم‌افزاری این مشکل مطابق زیر است:

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

    if (HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)) {
        HAL_Delay(100);

        HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_12);
        HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_13);
        HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_14);
        HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_15);
        while(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0));
        HAL_Delay(100);
    }
}
```

در روش فوق قراردادن تاخیرها منجر به از بین رفتن پدیده فوق خواهد شد.