آزمایش 5

طاها موسوی 98243058

نیلوفر مرادی جم 97243063

گروه 2

سوالات تحلیلی:

1 – اگر تنظیمات میکرو خود را در میان کار تغییر دهیم، هنگام تولید مجدد کد در STM32CubeMX چگونه میتوان از از بین رفتن کدهای توسعه داده شده خود جلوگیری کنیم؟

جواب: کد ها را بین USER CODE BEGIN و USER CODE END می گذاریم. سپس باید تعیین کنیم که این قسمت کد نگه داری یا بازنویسی شود. برای همین کار باید در قسمت Code Generator setting باید تعیین شود که کد بین دو کامنت گفته شده قرار گرفته است.

به این صورت می‌توانیم تنظیمات را حین کار تغییر دهیم بدون این که مشکلی پیش بیاید. برای مثال به این صورت:

```
/* USER CODE BEGIN 3 */
  HAL_GPIO_WritePin(GPIOI, GPIO_PIN_1, GPIO_PIN_RESET);
  HAL_Delay(1000);
  HAL_GPIO_WritePin(GPIOI, GPIO_PIN_1, GPIO_PIN_SET);
  HAL_Delay(500);
}
/* USER CODE END 3 */
```

2 – واژه کلیدی __weak در زبان C به چه معنی است و چرا برخی توابع CMSIS و HAL اینگونه تعریف شده اند؟

از این واژه کلیدی برای تعریف کردن توابع و vector tables و default handlers و توابع وقفه ها یا اعلان توابع و متغیر ها استفاده می‌شود. و باعث می‌شود کامپایلر نماد ها را ضعیف ارسال کند.

اگر تابعی از این نوع باشد، حتی در صورتی که از قبل در آن کدی نوشته شده باشد، قابلیت این را دارد تا برنامه نویس دوباره آن تابع را در فایل main بدون پیشوند weak بازنویسی کند.

اما اگر تابع از نوع strong باشد، قابلیت این را ندارد که بیشتر از یک بار اعلان شود. که در این صورت باید از weak کمک گرفت.

اما در صورت تعریف چندین weak، لینکر خطا می‌دهد. برای رفع این مشکل باید از گزینه muldefweak استفاده شود.

3 – جهت به کارگیری وقفه ها در HAL چه سازوکاری تعبیه شده؟ چگونه باید از آن بهره برد؟ ارتباط آن با سازوکار تعبیه شده در CMSIS چیست؟

در HAL به صورت کال بک این وقفه ها صدا زده می‌شوند و نیازی به interrupt service routine نیست. زمانی که interrupt handler در سطح CMSIS فعال می‌شود این متد ها هنگام اینتراپت در HAL صدا زده می‌شود. چون این متد ها یک لایه بالاتر از interrupt service routine ها هستند و کار کردن با آن ها ساده تر است.

ابتدا کار هایی مانند clear pending bits و .. را انجام می‌دهد و بعد آنها متد کال بک اجرا می‌شود. پس می‌توان کد های خود را درون متد کال بک نوشت تا اجرا شوند.

برای مثال در HAL می‌توان از متد EXTIx_HNADLER برای صدا زدن متد هندلر استفاده مرد و پین مربوط به آن Interrup line را مشخص کرد. و به کمک متدی که در HAL موجود است کار هایی که گفته شد و نیاز است را اجرا می‌کند و بعد آن متد کال بک اجرا می‌شود.
مشابه همین در CMSIS از همین متد به عنوان یک متد هندلر برای interrupt service مربوط به EXTIx استفاده می‌شود.

رفرنس های سوالات تحلیلی:

- کلاس درس و اسلاید های درسی

STM32Cube-MX-HAL-MOOC

https://melec.ir/%d8%b1%d9%88%d8%b4%d9%86-%da%a9%d8%b1%d8%af%d9%86-led-%d8%a8%d8%a7-%d9%85%db%8c%da%a9%d8%b1%d9%88-stm32/

https://stackoverflow.com/questions/35507446/what-are-weak-functions-and-what-are-their-uses-i-am-using-a-stm32f429-micro-co

https://sisoog.com/2019/01/06/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D9%85%DB%8C%DA%A9%D8%B1%D9%88%DA%A9%D9%86%D8%AA%D8%B1%D9%84%D8%B1-stm32f4-%D9%82%D8%B3%D9%85%D8%AA-%D8%B3%D9%88%D9%85-%D8%A7%DB%8C%D8%AC%D8%A7%D8%AF-%D9%BE%D8%B1/

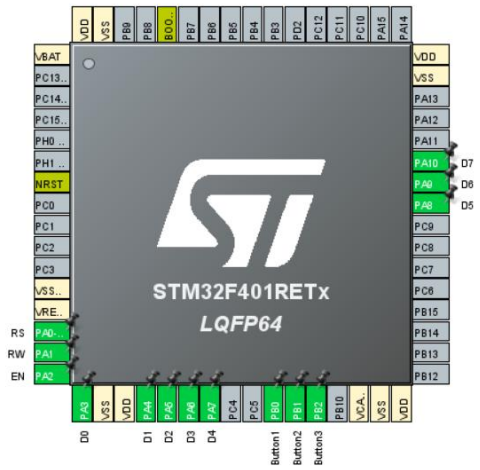
http://freeelec.ir/?p=454

دستور کار:

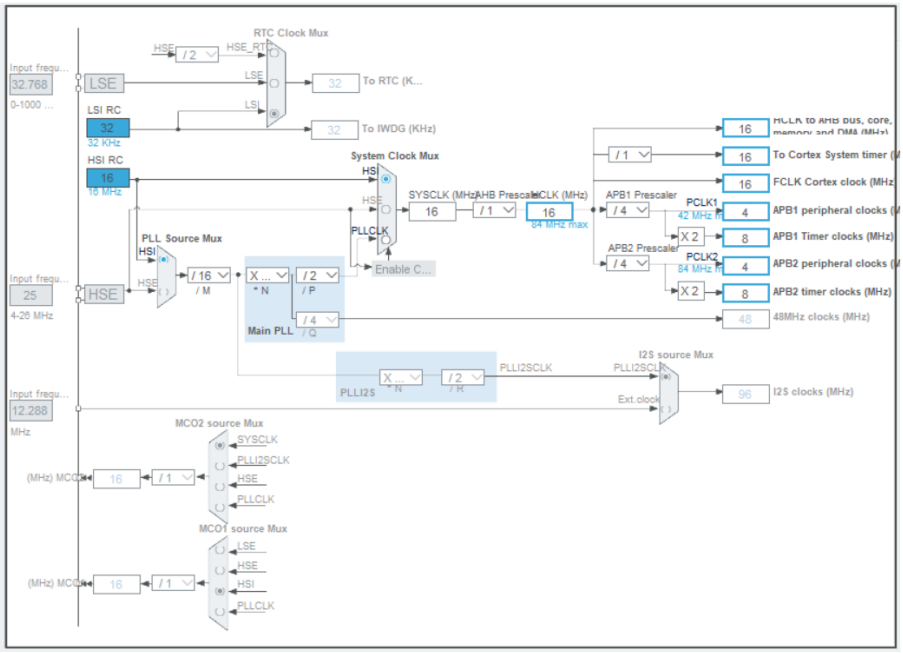
پورت ها را به صورت زیر کانفیگ میکنیم:

PA3 تا PA13 خروجی و متصل به lcd هستند.PA0 و PA1 و PA2 خروجی و متصل به پورت های EN RS RW هستند.

PB0و PB1 و PB2 به عنوان EXTI0_GPIO, EXTI1_GPIO, EXT12_GPIO ست میشوند و متصل به سه دکمه شماره یک و دو و سه هستند.



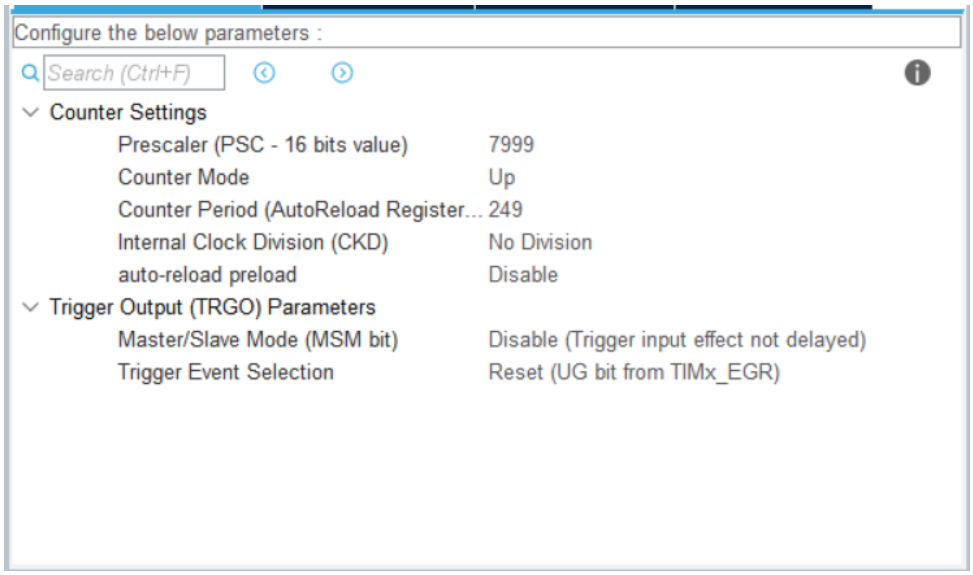
کالک تایمر 2 را که از باس APB1 تامین میشود را به صورت زیر تنظیم میکنیم:

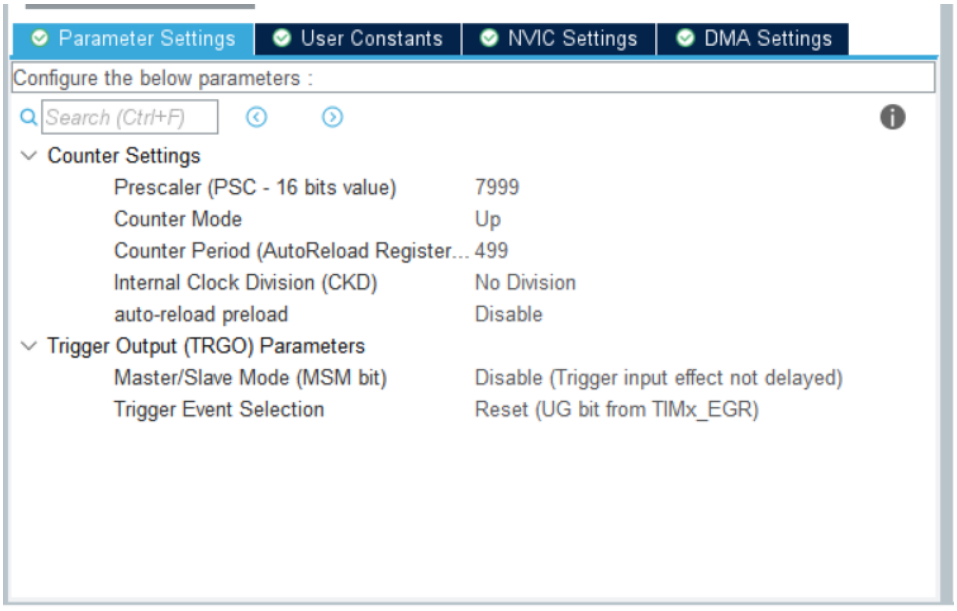


تایمر باید با فاصله های زمانی یک میلی ثانیه ای شمارش را انجام دهد. در تایمر ما از دو رجیستر prescaler و period counter برای تنظیم کردن این فاصله های زمانی هم استفاده میکنیم:

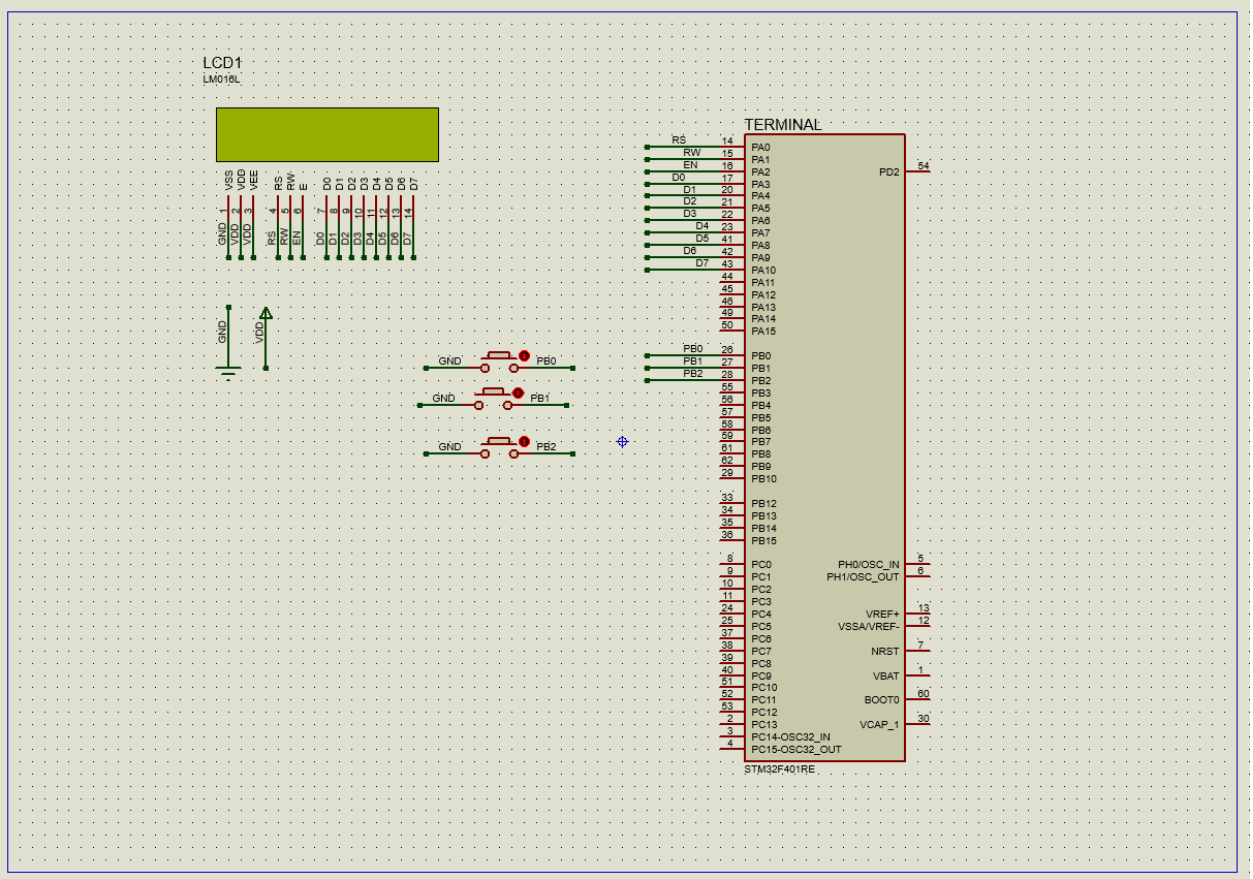
frequency = clock of timer / (prescaler * counter period)

که با توجه به اینکه کلاک تایمر 16 میباشد باید برای تولید بازه های یک میلی ثانیه ای، به عنوان مثال، prescalerبرابر 499 و period counterبرابر 249 باشد تا تغییر هر 250 میلی ثانیه یک بار باشد





پروتئوس را مشابه بورد stm cubex طراحی میکنیم.



در ادامه با جنریت شدن کد توسط cubemx کد را طبق منطق صورت پروژه تغییر میدهیم.

```
344
345  // To Be Called When One Of The Buttons Pushed
346  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
347    // When First Button Clicked
348    if(GPIO_Pin == Button1_Pin){
349      // When The State Is not in Turn Off Mode
350      if(isTurnOff == 0){
351      // When First Button Pushed
352      if(GPIOB->IDR){
353      // Start TIM3 To Counting
354      HAL_TIM_Base_Start_IT(&htim3);
355      } }}
356      // When Second Button Clicked
357    else if(GPIO_Pin == Button2_Pin){
358          // When The State Is not in Turn Off Mode
359        if(isTurnOff == 0){
360              // When Second Button Pushed
361      if(GPIOB->IDR){
362            // Stop TIM3
363      HAL_TIM_Base_Stop_IT(&htim3);
364      } }}
365        // When Third Button Clicked
366    else if (GPIO_Pin == Button3_Pin){
367      HAL_NVIC_SetPendingIRQ(EXTI2_IRQn);
368      // Stop TIM3
369      HAL_TIM_Base_Stop_IT(&htim3);
370      // Reset Timer To 00:00:000
371      ResetTimer();
372      int pushedFor3000ms = 1;
373      // Get Current HAL_Tick In ms
374      uint32_t start = HAL_GetTick();
375      // Wait For 3 seconds
376      while((HAL_GetTick()- start)<500){
377          // No Need For Switching To Turn Off Mode If Release The Third Button
378          if(HAL_GPIO_ReadPin(GPIOB, MASK(2) ) != GPIO_PIN_SET){
379            pushedFor3000ms = 0;
```

بعد از تغییر در کلید متود بالا فراخوانی شده و با توجه به این که تغییر در کلید اول یا دوم یا سوم بوده است و طبق منطق صورت سوال تغییرات لازم اعمال میشود و خروجی نمایش داده میشود.

تایمر هر 250 میلی ثانیه یک بار آپدیت شده و زمان جدید روی lcd نمایش داده میشود.

```
315
316 // To Be Called When TIM3 Overflows (250 ms)
317 void TIM3_IRQHandler(void){
318       // Reset Cursor To The Beginning Of The Line
319       ResetCursor();
320       // Increment Timer  +250 ms
321       IncrementTime();
322       // Print Time
323       ShowTimer();
324
325 }
```

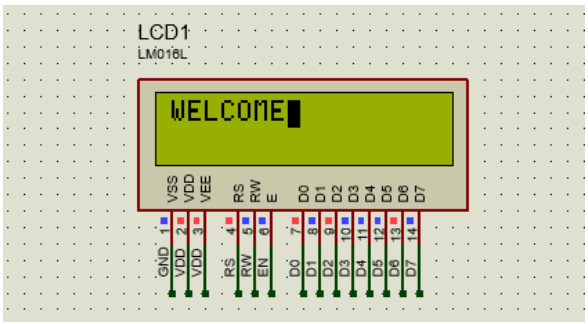
```
325 }
326 // To Be Called When TIM4 Overflows (500 ms)
327 void TIM4_IRQHandler(void){
328   // Clear Line
329   ClearLine();
330   // Reset Cursor To The Beginning Of The Line
331   ResetCursor();
332   // Delay For Blinking
333   Delay(10);
334   // Print Turn Off
335   EnterLCDData('T');
336   EnterLCDData('U');
337   EnterLCDData('R');
338   EnterLCDData('N');
339   EnterLCDData(' ');
340   EnterLCDData('O');
341   EnterLCDData('F');
342   EnterLCDData('F');
343 }
```

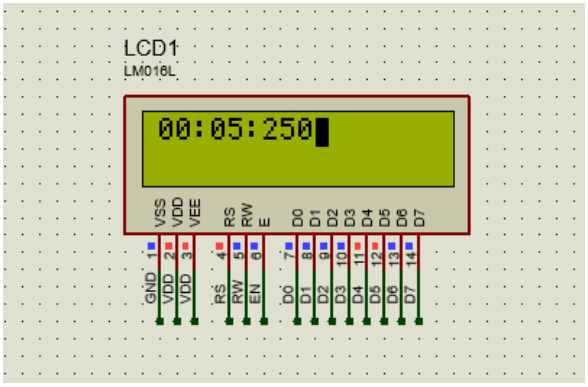
به سراغ Callback های مربوط به تایمر ها می رویم. از آنجاکه Callback مربوط به تایمر یعنی PeriodElapsedCallBack_TIM_HALدر هنگام سرریز شدن تایمرها صدا زده نمیشد, از Handler Interruptهای پیش فرض مربوط به این دو تایمر استفاده کرده ایم.

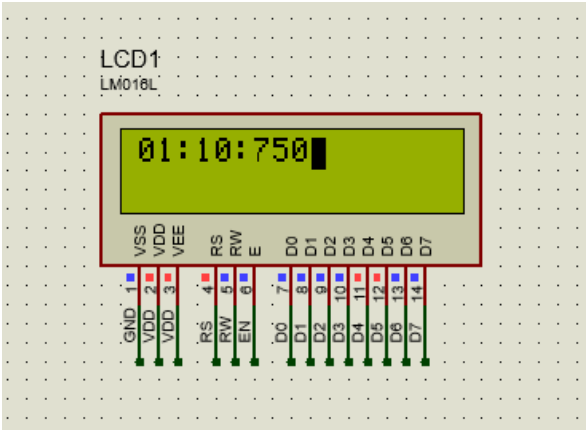
برای نمایش از زمان نری متد ShowTimer و با استفاده از متغریهای تعریف شده, به صورت زیر استفاده میکنیم تا به فرمت mmm:SS:MM نمایش داده شود.

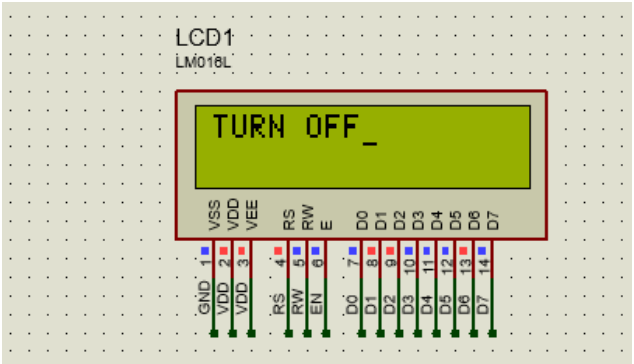
```
485 // Show Timer In 00:00:000 Format
486 void ShowTimer(void){
487   ShowNumber(minute/10);
488   ShowNumber(minute%10);
489   EnterLCDData(':');
490   ShowNumber(second/10);
491   ShowNumber(second%10);
492   EnterLCDData(':');
493   ShowNumber(milisecond/100);
494   ShowNumber(milisecond%100 / 10);
495   ShowNumber(milisecond%10);
496 }
497
498
499 void ShowNumber(int input){
500   // Enter 0 To LCD
501   if(input == 0){
502     EnterLCDData('0');
```

در نهایت مشاهده خروجی در پروتئوس:









رفرنس دستور کار:

کلاس درس و اسلاید های درسی

دیتا شیت و رفرنس منوآل

https://www.keil.com/pack/doc/cmsis/DSP/html/index.html

https://www.st.com/en/embedded-software/stsw-stm32065.html

https://www.keil.com/dd/docs/arm/st/stm32f4xx/stm32f4xx.h

https://www.keil.com/dd/docs/arm/st/stm32f4xx/stm32f4xx.h