GÖREV BİLDİRİMİ (TASK NOTIFICATION) NEDİR?

RTOS (Real-Time Operating System) içinde, "Task Notification" bir görevin diğer görevler tarafından haberdar edilmesi anlamında kullanılır. Bu haberleşme, görevler arasında iletişimi ve koordinasyonu sağlamak için kullanılabilir. Task Notification, bir görevin tamamlanması, beklenen bir durumun oluşması veya başka bir görevin belirli bir durumda haberdar olması gibi durumlar için kullanılabilir. Task Notification mekanizması, RTOS tarafından sağlanan bir özelliktir ve görevlerin daha verimli bir şekilde çalışmasını ve birbirleriyle etkileşimini sağlar.

RTOS' da her bir görevin bir dizi görev bildirimi vardır bu görevlerin **"beklemede"** veya **"beklemede değil"** olabilen bildirim durumu ve 32 bitlik bir bildirim değeri vardır vebu değer 0 olarak başlatılır.

Görev bildiriminde xTaskNotify() ve xTaskNotifyWait() olmak üzere kullanılan iki adet API mevcuttur.

xTaskNotifyWait() Bu API' yi çağıran görev bir olay bekler.

xTaskNotify() bu API' yi çağıran görev, görev bildirimi bekleyen göreve bir bildirim gönderir

xTaskNotifyWait() API' SI

task.h başlık dosyasında bildirilmiştir.

4 adet parametresi vardır

BaseType_t xTaskNotifyWait(uint32_t ulBitsToClearOnEntry,

uint32_t ulBitsToClearOnExit,

uint32_t *pulNotificationValue,

TickType_t xTicksToWait);

Çağıran görevin bir bildirim alması için isteğe bağlı bir zaman aşımı ile bekler. Beklerken CPU döngülerini tüketmez. Başka görevlerden veya kesme işleyicisinden bir bildirim alana kadar isteğe bağlı bir zaman aşımı ile bekler. Böylece kesme işleyicisinden gübenli olan FreeRTOS API' lerini kullanarak kesme işleyicisinden de bildirim gönderebiliriz

1. **PARAMETRE uint32_t ulBitsToClearOnEntry:**

bu barametrede ayarlanan bitler, xTaskNotifyWait() çağırıldığında bir bildirim beklemede değilse, xTaskNotifyWait() işlevine girişte (görev yeni bir bildirim beklemeden önce) çağıran RTOS görevinin bildirim değeri temizlenir.
örneğin Örneğin ulBitsToClearOnEntry değeri 0x0000 0001 ise, görevin bildirim değerinin 0. Biti dolu demektir ve bildirim değerinin 0. Biti işleve girişte temizlenir. Eğer 0xFFFF FFFF olarak ayarlanmışsa, o görev bildirim değerindeki tüm bitleri temizler.

Kısaca bu API çağırıldığında o görevin bildirim değeri sıfırlanır.

2. **PARAMETRE uint32_t ulBitsToClearOnExit**

Bildirim alınıp task içindeki işlemler yerine getirildikten sonra task görevinden hemen çıkmadan önce çağırılan RTOS görevinin bildirim değerini sıfırlar. RTOS görevinin bildirim değeri *pulNotificationValue içine kaydedildikten sonra bitler temizlenir.

Örneğin ulBitsToClearOnExit 0x03 ise yani görevin 0. ve 1. Bitleri dolu ise görevin bildirim değerinin 0 ve 1. Bitleri işlemden çıkılmadan önce silinir eğer ulBitsToClearOnExit 0xFFFF FFFF ise tüm bitleri temizlenir.

3. **PARAMETRE uint32_t *pulNotificationValue**

RTOS bildirim değerini iletmek için kullanılır bu parametreye kopyalanan değer ulBitsToClearOnExit parametresi nedeniyle herhangi bir bit temizlenmeden önceki haliyle RTOS görevinin bildirim değeridir eğer bu değer gerekli değilse **NULL** olarak ayarlanabilir

4. **PARAMETRE TickType_t xTicksToWait**

xTaskNotifWait() çağırıldığında bir bildirim beklemede değilse, bir bildirim alınması için **"ENGELLEME"** Durumunda beklenecek en uzun süreyi belirtir.

Bu süre ms cinsinden değil Tick cinsindendir (Not: RTOS API'lerinin argümanları asla ms, us, ns cinsinden değer almazlar, her zaman RTOS Tick cinsinden değer alırlar)

NOT: Zaman RTOS Tick periyotunda belirtilir. Eğer ms cinsinden belirtmek istiyor isek RTOS tarafından tanımlanan **pdMS_TO_TICK()** makrosunu kullanmalıyız. Bu metot milisaniyeyi RTOS Tick değerine dönüştürür.

Eğer xTaskNotifyWait() çağırıldığında bir bildirim alınmışsa veya bir bildiirm beklemedeyse pdTrue döner.

xTaskNotifyWait() çağırıldığında bir bildirim alınmadan önce zaman aşımına uğrarsa pdFalse döner.

xTaskNotify() API' SI

task.h başlık dosyasında bildirilmiştir.
BaseTypeNotify(TaskHandle_t xTaskToNitify,
Uint32_t ulValue,
eNotifyAction eAction);
Bu API doğrudan bir RTOS görevine göndermek ve potansiyel olarak engelini kaldırmak isteğe bağlı olarak aşağıdaki yollardan biriyle alıcı görevin bildirim değerini güncellemek için kullanılır.

- Bildirim değerine 32 bitlik bir sayı yazmak için
- Bildirim değerini 1 artırmak için
- Bildirim değerine bir veya daha fazla bit ayarlamak
- Bildirim değerini değiştirmeden bırakmak için kullanılır

Bu API bir kesme fonksiyonu (ISR) içerisinden çağırılmamalıdır bunun yerine **xTaskNotifyFromISR()** kullanılmalıdır.

1. **PARAMETRE TaskHandle_t xTaskToNitify**

Bu parametre RTOS görevinin tanıtıcısıdır. Bir görevin tanıtıcısını elde etmek için **xTaskCreate()** fonksiyonu ile görev oluşturulur ve **pxCreatedTask** parametresi kullanılır yada **xTaskCreateStatic()** ile görev oluşturulur ve döndürülen değer saklanır yada **xTaskGetHandle()** ile görevin adı kullanılır.

2. **PARAMETRE uint32_t ulValue**

Bu parametre RTOS görevinin bildirim değerini güncellemek için kullanılır.

3. **PARAMETRE eNotifyAction eAction**

Bu parametre ilişkili eylemi gerçekleştirmek için aşağıdaki belirtilen değerlerden birini alabilen enum türüdür.

eNoAction:

Hedef görev, olayı alır, ancak bildirim değerini güncellemez. Bu durumda ulValue kullanılmaz

eSetBits:

Hedef görevin bildirim değeri, ulValue ile bit düzeyinde ORed olacaktır. Örneğin ulValue 0x01 olarak ayarlanırsa, hedef görevin bildirim değeri içinde bit 0 olarak ayarlanır. Benzer şekilde, ulValue 0x04 ise, hedef görevi bildirim değerinde bit2 ayarlanacaktır. Bu şekildeRTOS görev bildirim mekanizması, bir olay grubuna hafif ir alternatif olarak kullanılabilir.

eIncrement:

Hedef görevin bildirim değeri bir artırılarak **xTaskNotify()** çağırısı, **xTaskNotfyGive()** çağırısına eşdeğer hale getirilir. Bu durumda ulValue kullanılmaz.

eSetValueWithOverwrite:

Hedef görevin hazırda bekleyen bir bildirimi olsun ya da olmasın bildirim değeri ulValue olarak ayarlanacaktır. Bu şekilde RTOS görev bildirim mekanizması, xQueueOverwrite()' a hafif bir alternatif olarak kullanılmaktadır.

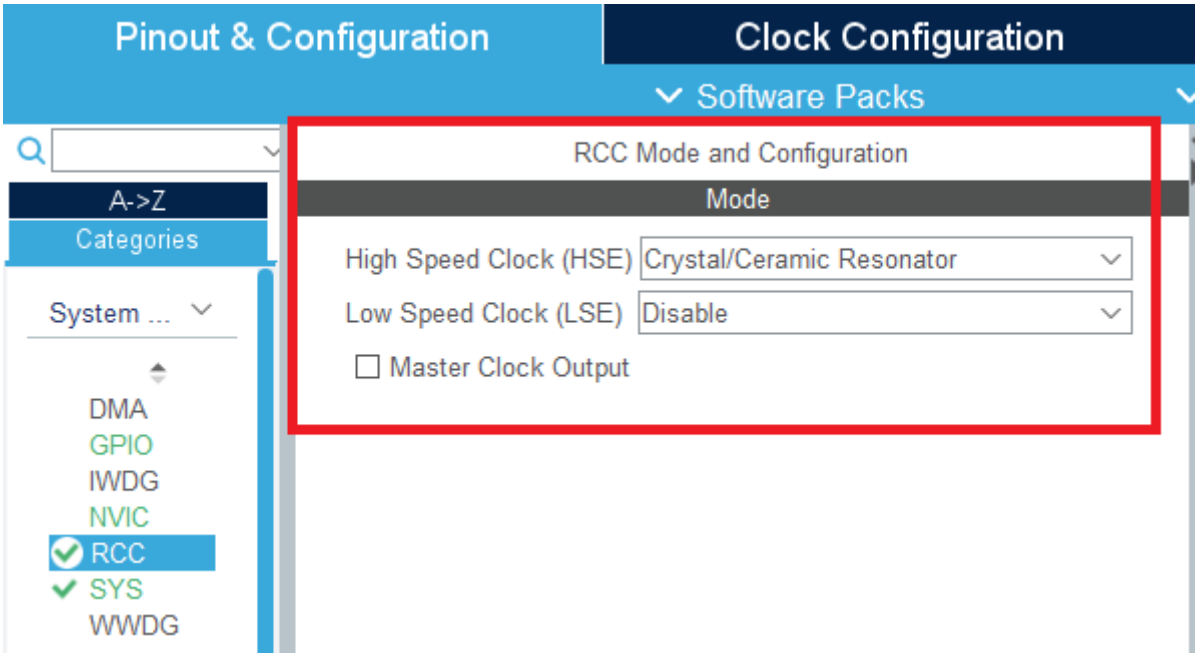
eSetValueWithoutOverwrite:

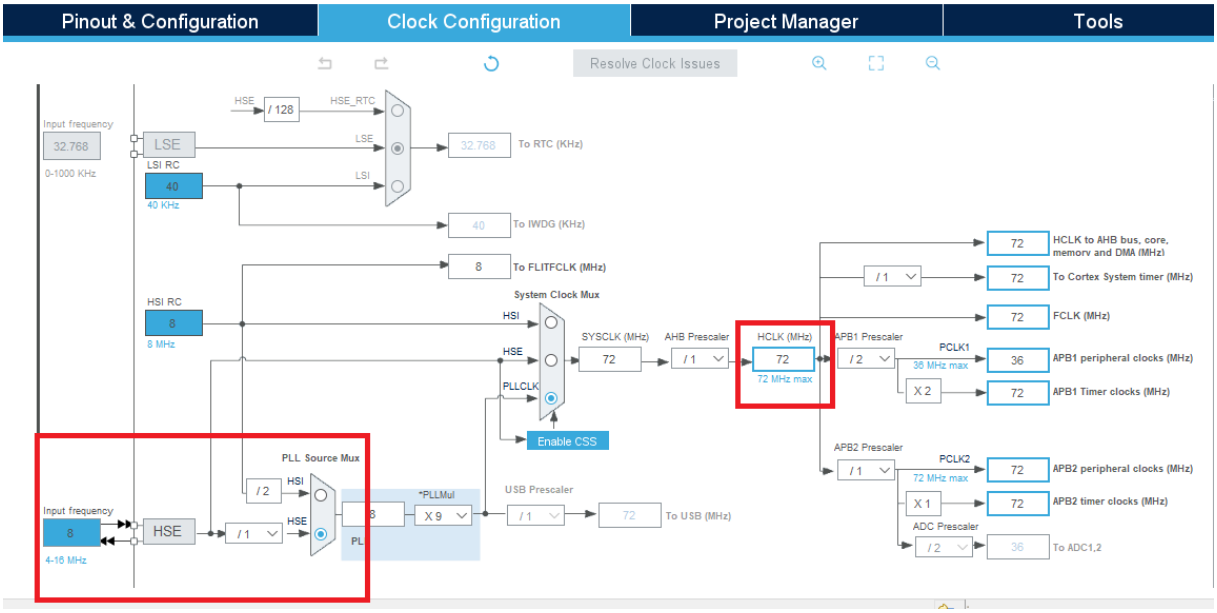
Hedef görevin hazırda bekleyen bir bildirimi yoksa bildirim değeri ulValue olarak ayarlanacaktır. Hedef görevin halihazırda bekleyen bir bildirimi varsa, bildirim değeri güncellenmez, böylece kullanılmadan önce önceki değerin üzerine yazılır bu durumda xTaskNotify() çağırısı başarısız olur ve pdFALSE döndürülür.

FreeRTOS' DA BİR GÖREVDEN BAŞKA BİR GÖREVİ AKTİFLEŞTİRME (TASK NOTIFICATION) UYGULAMASI

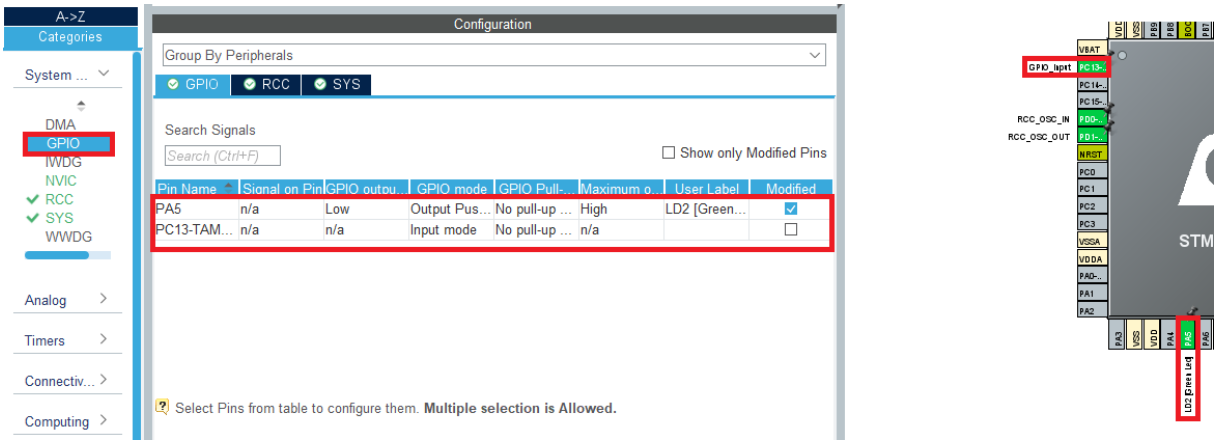
Bu projede 2 adet görev oluşturacağız bu görevlerden biri buton okurken diğer görev butona her basıldığında kartın üzerinde bulunan ledi yakıp söndürecek.

İlk olarak bu projede NUCLEO-STM32F103RB kartımızı kullanıyoruz projeyi oluşturup osilatör ve saat ayarlamalarını yapıyoruz. Saat hızımızı 72MHz, Osilatörümüzü ise kristal/seramik osilatör olarak seçiyoruz.

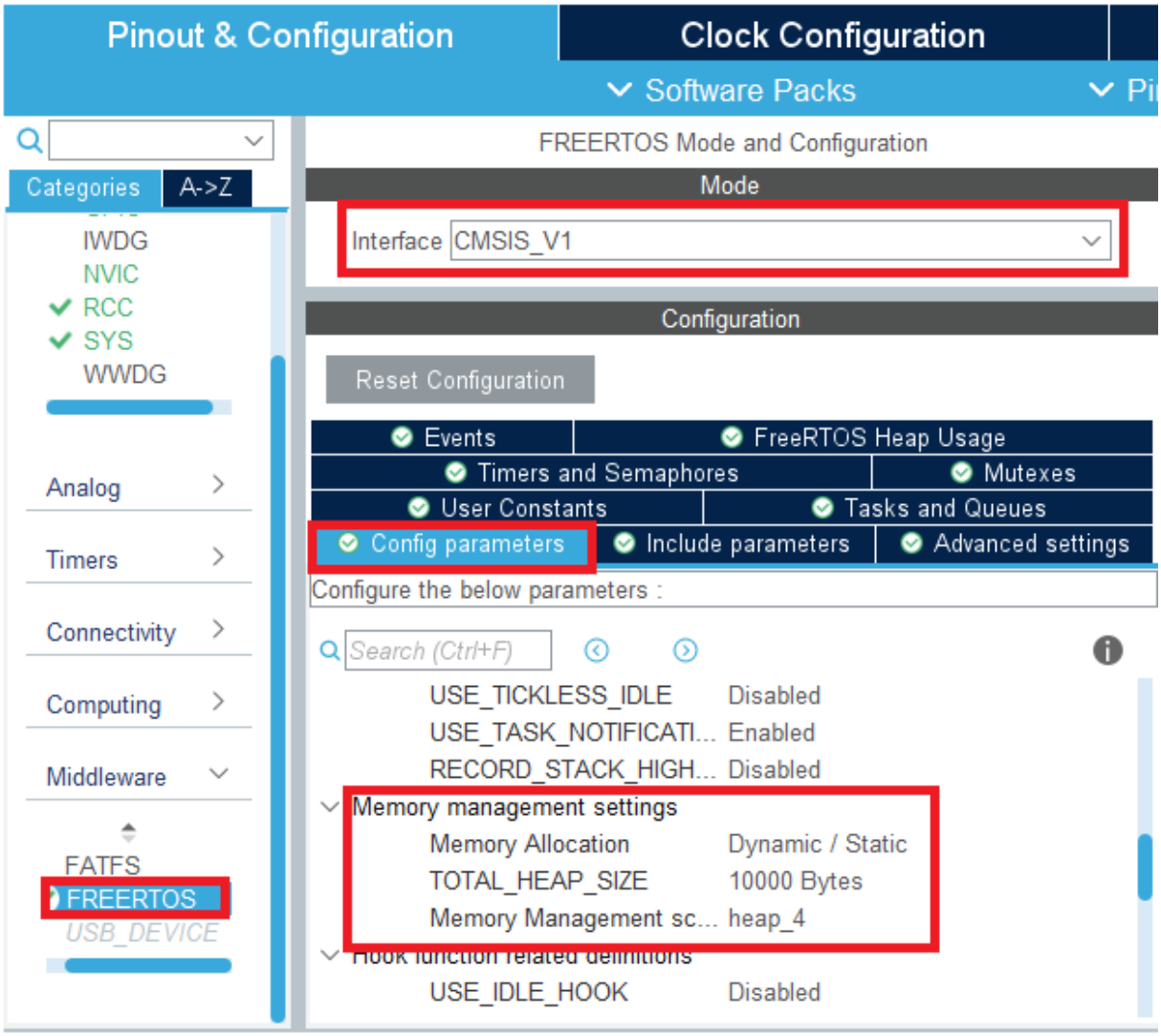




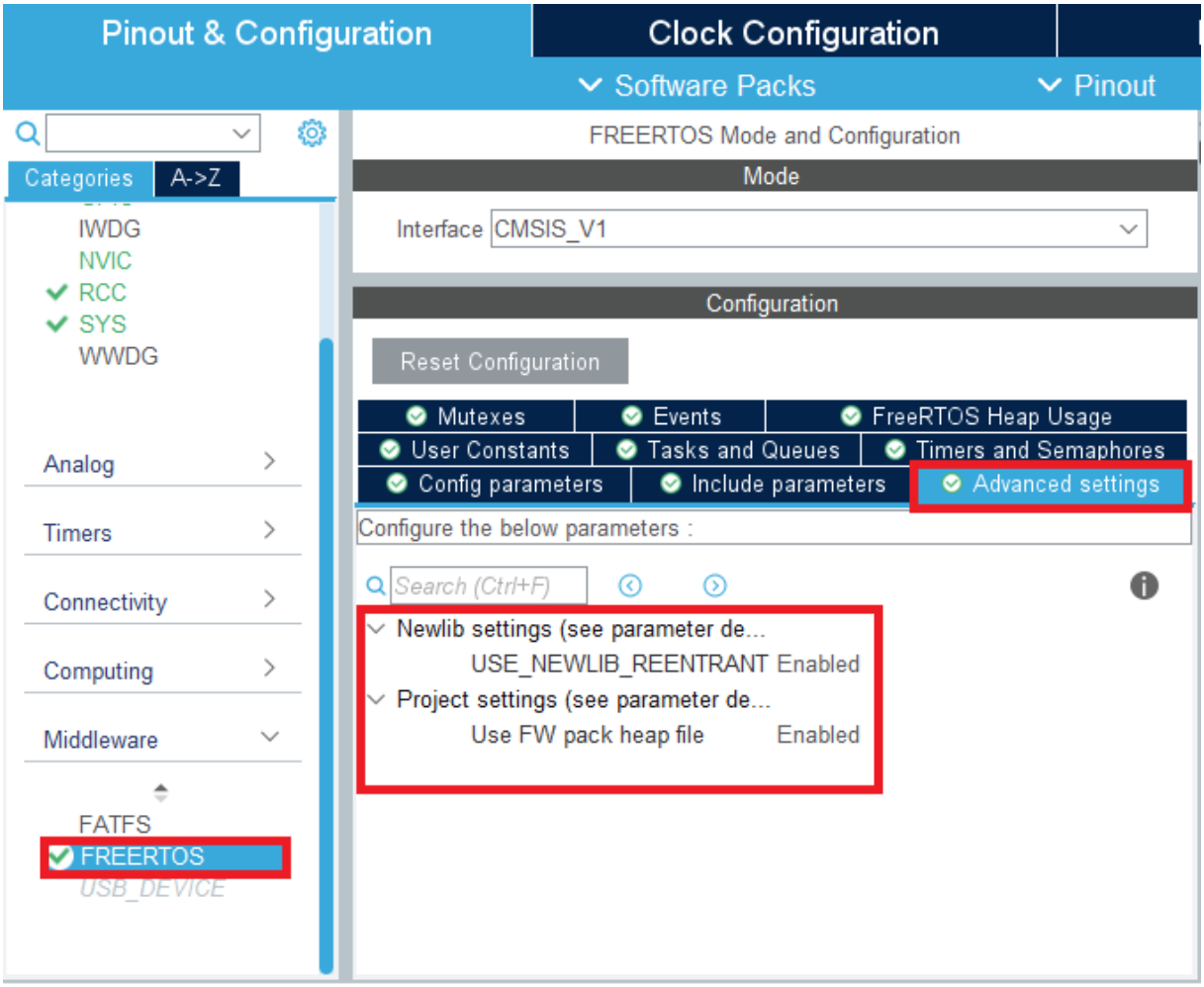
Şimdi GPIO ayarlamalarımızı yapacağız bu kartımızda PA5' e bağlı led ve PC13' e bağlı buton bulunmaktadır bundan dolayı PA5' i çıkış PC13' ü ise giriş olarak ayarlayacağız.



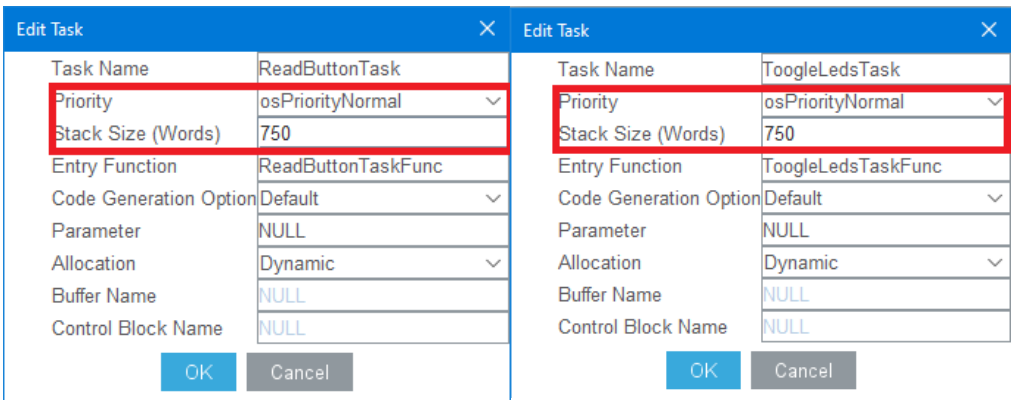
MiddleWare kısmında bulunan FreeRTOS' u aktifleştirip Config parameters kısmının altında bulunan Memory management settings kısmında TOTAL_HEAP_SIZE parametresini 10000 bayt olarak düzenliyoruz.



Advanced settings kısmında bulunan Newling settings kısmında bulunan USE_NEWLIB_REENTRANT parametresini **“ENABLED”** şeklinde düzenliyoruz.



Task and Queues kısmına gidip ReadButtonTask ve ToggleLedsTask adında 2 adet görev oluşturuyoruz



CTRL + S basıp STM32CubeMX’ in kodları oluşturmasını bekliyoruz ve ToggleLedsTaskFunc ve ReadButtonTaskFunc fonksiyonlarını aşağıdaki gibi dolduruyoruz.

```
/* USER CODE END Header_ReadButtonTaskFunc */
void ReadButtonTaskFunc(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      if(!HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13))
      {
          xTaskNotify(ToogleLedsTaskHandle,0x00,eNoAction);
          while(!HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13));
      }
  }
  /* USER CODE END 5 */
}
```

```
/* USER CODE END Header_ToogleLedsTaskFunc */
void ToogleLedsTaskFunc(void const * argument)
{
  /* USER CODE BEGIN ToogleLedsTaskFunc */
  /* Infinite loop */
    uint32_t NotficationValue = 0;
  for(;;)
  {
      if(xTaskNotifyWait(0, 0, &NotficationValue, portMAX_DELAY) == pdTRUE)
      {
          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
      }


  }
  /* USER CODE END ToogleLedsTaskFunc */
}
```

Burada beklemede olan fonksiyon ToogleLedsTaskFunc()' ta beklemeyi xTaskNotifyWait() fonksiyonu ile sağlıyoruz

NOT: portMAX_DELAY maksimum bekleme süresini ifade ediyor

Burada diğer fonksiyonu tetikleyen fonksiyon olan ReadButtonTaskFunc()' ta tetiklemeyi xTaskNotify () fonksiyonu ile sağlıyoruz

NOT: eNoAction parametresi karşı tarafın NotificationValue parametresinin sabit kalmasını sağlıyor

NOT: pdTRUE parametresi ise tetiklenme geldiğinde dönen parametredir.

Son olarak aşağıdaki github' dan projenin çalışma videosuna ulaşabilirsiniz.