HOOK FUNCTION NEDİR?

FreeRTOS'da Hook functions, sistem olaylarının veya diğer özel durumların yakalanması ve işlenmesi için kullanılan özel fonksiyonlardır. FreeRTOS, belirli durumlar için "hook points" tanımlar ve bu noktalarda çalıştırılacak olan hook fonksiyonlarını tanımlanmasına olanak tanır. Örneğin, FreeRTOS, bir task' ın başlamadan veya bitmeden önce veya bir kesme işlemesi sırasında hook fonksiyonları çalıştırmasına izin verir.

Hook fonksiyonları, FreeRTOS tarafından sağlanan varsayılan işlevselliğin üzerine ek özelleştirme yapmak için kullanılabilir. Örneğin, task switch olayı sırasında bir profil çizmek isteyebilirsiniz veya task' ın bitiş zamanını kaydetmek isteyebilirsiniz. Bu tür işlemler, varsayılan FreeRTOS fonksiyonalitesini değiştirmesine gerek kalmadan yapılabilir.

FreeRTOS' DA BULUNAN HOOK FUNCTIONS NELERDİR?

- IDLE Hook Function
- RTOS Tick Hook Function
- Dynamic Memory Allocation Failed Hook Function (Malloc Failed Hook Function)
- Static Overflow Hook Function

IDLE HOOK FUNCTION NEDİR?

IDLE hook function, bir işletim sistemi tarafından kullanılan bir yapıdır ve uygulamanın işlem yapmakta olduğu bir duraklama sırasında yapması gereken görevleri belirler. Genellikle, bir işletim sistemi boş vakti olduğunda, IDLE hook function çalıştırılır ve sistem görevleri gibi belirli görevleri yapmasına izin verir.

Örneğin, bir uygulama, IDLE hook function ile bellek yönetimini optimize etmek veya fonksiyonların güncellenmesini sağlamak için bu boş zamanı kullanabilir.

IDLE hook function, genellikle yüksek performanslı ve verimli işletim sistemleri için kullanılır ve uygulamanın verimliliğini ve performansını arttırmak için tasarlanmıştır.

RTOS TICK HOOK FUNCTION NEDİR?

RTOS (Real-Time Operating System) Tick hook function, RTOS'lar içinde zaman dilimlerinin (ticks) yönetilmesinde kullanılan bir fonksiyondur. Her zaman dilimi, RTOS'lar tarafından belirli bir periyodik frekans ile yapılan bir zaman ölçeğidir ve genellikle milisaniye veya mikrosaniye olarak belirlenir.

RTOS Tick hook function, RTOS'lar tarafından her zaman dilimi çalıştırılır ve bu fonksiyon, uygulamanın her zaman dilimi içinde yapması gereken görevleri belirler. Örneğin, bir uygulama, RTOS Tick hook function ile bellek yönetimini optimize etmek veya fonksiyonların güncellenmesini sağlamak için bu zaman dilimini kullanabilir.

RTOS Tick hook function, gerçek zamanlı uygulamalar için tasarlandığından, uygulamanın zaman dilimi boyunca verimli ve doğru bir şekilde çalışmasını sağlamak için tasarlanmıştır. Bu fonksiyon, RTOS'lar içinde öncelikli olarak kullanılır ve uygulamanın performansını ve verimliliğini arttırmak için önemlidir.

MALLOC FAILED HOOK FUNCTION NEDİR?

Malloc Failed hook function, bellek yönetimi için kullanılan bir fonksiyondur ve bellek atama işlemi başarısız olduğunda çalışır. Malloc (Memory Allocation) fonksiyonu, bir uygulamanın bellekte bellek alanı istemesini sağlar ve genellikle dinamik bellek yönetimi için kullanılır.

Eğer bellek atama işlemi başarısız olduğunda, Malloc Failed hook function çalışır ve bu fonksiyon, uygulamanın bellek yetersizliği durumunda ne yapması gerektiğini belirler. Örneğin, bir uygulama, Malloc Failed hook function ile bellek tasarrufu yapabilir veya bellek hatasını raporlayabilir.

Malloc Failed hook function, bellek yönetimi için öncelikli olarak kullanılan ve uygulamanın bellek yetersizliği durumunda doğru bir şekilde yanıt vermesini sağlamak için tasarlandı. Bellek yetersizliği durumunda, uygulamanın hata vermemesi ve doğru bir şekilde yanıt vermesi için, Malloc Failed hook function çok önemlidir.

STATIC OVERFLOW HOOK FUNCTION NEDİR?

Static Overflow hook function, bellek yönetimi için kullanılan bir fonksiyondur ve bellek depolama alanının sınırlarının aşılması durumunda çalışır. Static (sabit) bellek, bir uygulamanın çalışma sırasında kullanması gereken bellek miktarını önceden belirleyen ve bu bellek miktarı uygulamanın sonunda boşaltılan bellektir.

Eğer bir uygulama, sabit bellek depolama alanını aşarsa, Static Overflow hook function çalışır ve bu fonksiyon, uygulamanın bellek depolama alanının sınırlarının aşılması durumunda ne yapması gerektiğini belirler. Örneğin, bir uygulama, Static Overflow hook function ile bellek depolama alanı hatasını raporlayabilir veya uygulamanın sonlandırılmasını sağlayabilir.

Static Overflow hook function, bellek yönetimi için öncelikli olarak kullanılan ve bellek depolama alanının sınırlarının aşılması durumunda doğru bir şekilde yanıt vermesini sağlamak için tasarlandı. Bellek depolama alanının sınırlarının aşılması durumunda, uygulamanın hata vermemesi ve doğru bir şekilde yanıt vermesi için, Static Overflow hook function çok önemlidir.

WFI, WFE TALİMATLARI NEDİR?

STM32 microcontrollers'deki "WFI" ve "WFE" talimatları, microcontroller'daki işlemcisinin bekleme (wait) modunda olduğunu belirtir. Bu talimatlar, güç tasarrufu yapmak ve işlemci gücünü minimum düzeyde tutmak amacıyla kullanılır.

"WFI" talimatı, işlemciyi bekleme durumuna alır ve bekleme durumundaki işlemci, bir kesme oluşana kadar aktif hale gelmez.

"WFE" talimatı ise, bekleme durumundaki işlemciye bir kesme oluşana kadar uyanması için gerekli olan bir öncelik sırası belirler. Bu talimat, öncelikli kesmeleri işlemenize olanak tanır ve diğer kesmeleri beklemek zorunda bırakır.

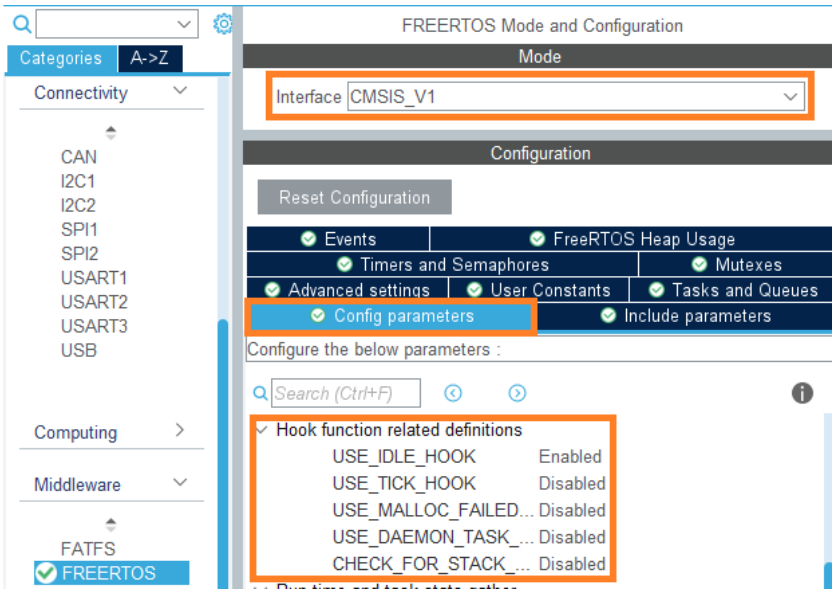
Genel olarak, bu talimatlar, güç tasarrufu sağlamak ve işlemci performansını iyileştirmek için kullanılır. Aynı zamanda, sistemi uyku durumuna almak için de kullanılabilir, ancak daha fazla güç tasarrufu sağlamak amacıyla genellikle "sleep" modu tercih edilir.

KANCA İŞLEVİ İLE MİKRODENETLEYİCİYİ UYKUYA SOKMA PROJESİ

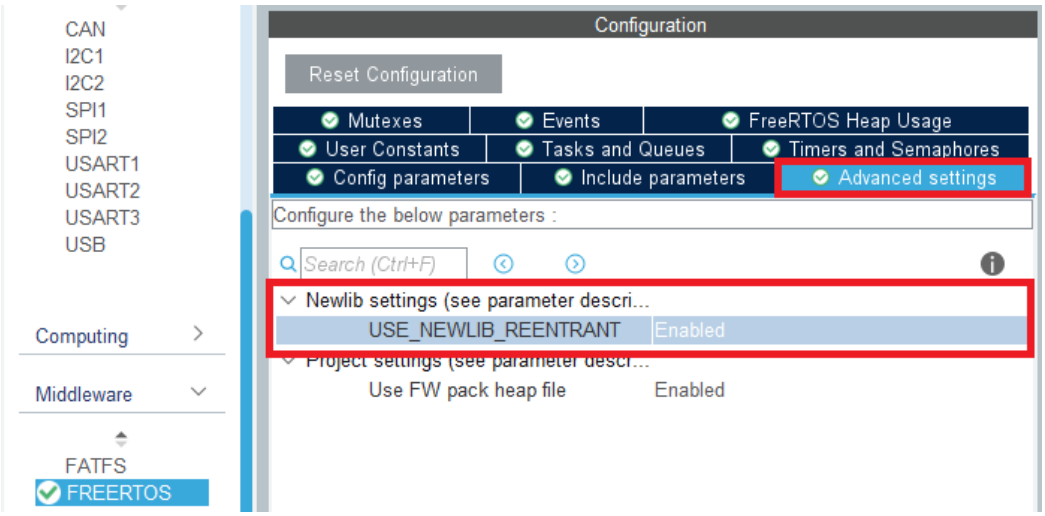
Bu projede PA5 pinimize bağlı led 100 ms' de bir toogle yaptırılacak IDLE TASK HOOK fonksiyonunda ise mikrodenetleyici __WFI(); talimatı ile uyku durumuna alınacak.

Öncelikle Pinout & Configuration kısmında bulunan LSE kısmını DISABLE, HSE kısmını ise Crystal/Ceramic Resonator olarak düzeltiyoruz ardından ardından PA5 pinimizi çıkış olarak ayarlıyoruz.

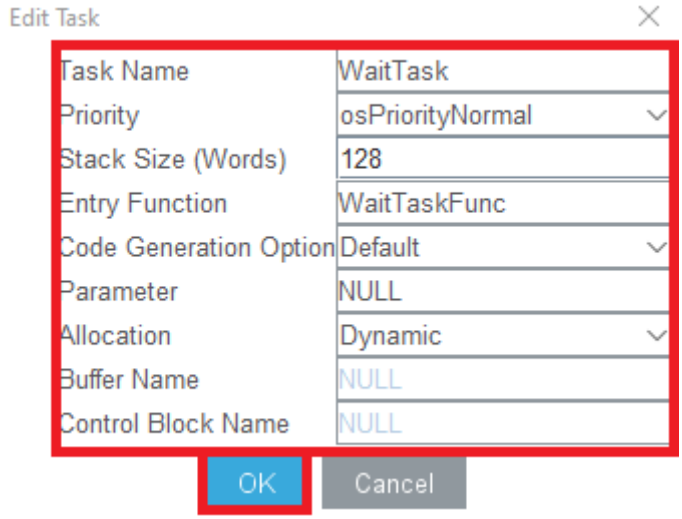
Middleware kısmında bulunan FreeRTOS' u aktifleştirif Interface kısmını CMSIS_V1 olarak düzenliyoruz ve config parameters kısmında "Hook functions related definitions" kısmınının altında bulunan **"USE_IDLE_HOOK"** parametresini ENABLED olarak düzenliyoruz.



Daha sonra Config parameters'in üstünde bulunan Advanced settings bölümünde yer alan **USE_NEWLIB_REENTRANT** parametresini **ENABLED** olarak düzenliyoruz.



Şimdi yeni bir görev oluşturuyoruz.



WaitTaskFunc() Fonksiyonunun içini aşağıdaki gibi dolduruyoruz.

```
/* USER CODE END Header_WaitTaskFunc */
void WaitTaskFunc(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    osDelay(1000);
  }
  /* USER CODE END 5 */
}
```

Daha sonradan vApplicationIdleHook() fonksiyonunu main.c içinde tanımlayıp içini dolduruyoruz.

```
/* USER CODE BEGIN 0 */
void vApplicationIdleHook(){
    __WFI();
}
/* USER CODE END 0 */
```

Her vApplicationIdleHook() fonksiyonuna girdiğinde led' e toogle yaptırılacak ve uyku moduna girecek (WaitTaskFunc fonksiyonuna girene kadar uyku modunda kalacak)