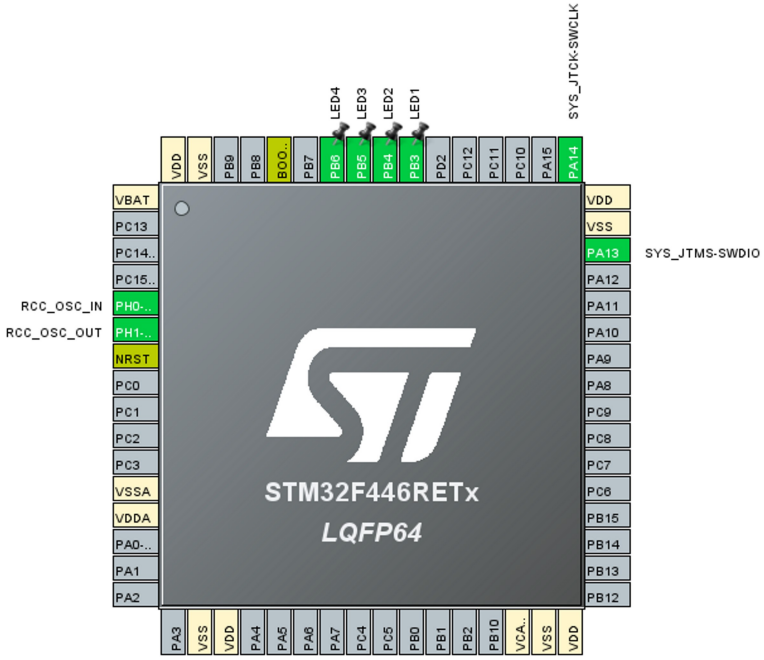
Timer Interrupt

25 Aralık 2021 Cumartesi 00:59

Timer Interrupt

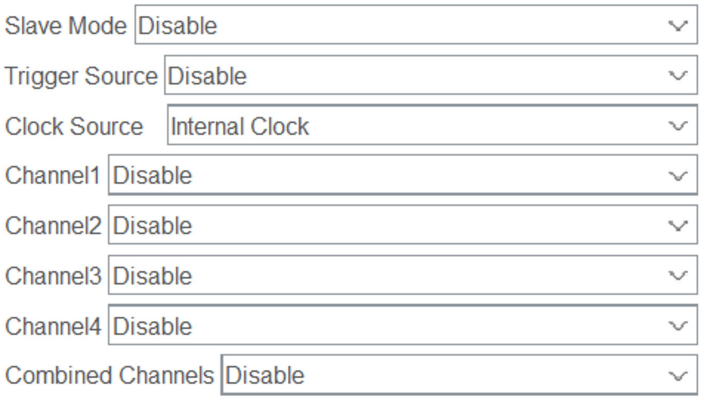
- HAL

Konfigürasyon Kısmı

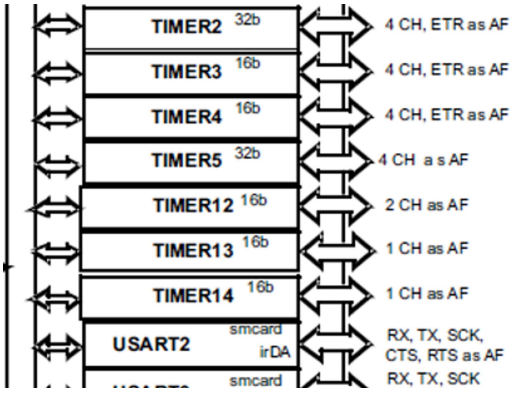


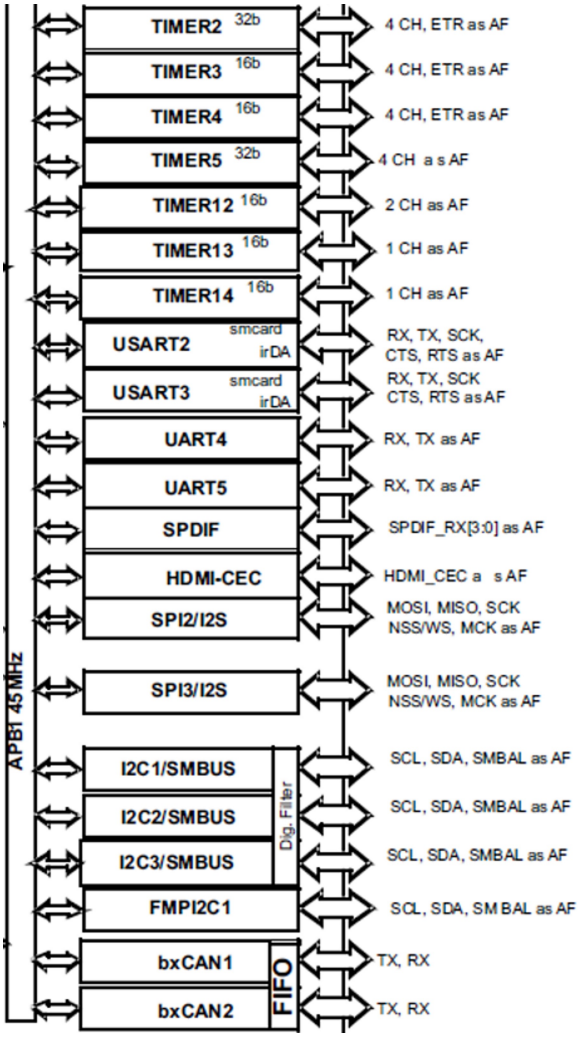
Pin N...	Signal on ...	GPIO out...	GPIO mode	GPIO Pull...	Maximum ...	User Label	Modified
PB3	n/a	Low	Output Pu...	No pull-up...	Very High	LED1	☑
PB4	n/a	Low	Output Pu...	No pull-up...	Very High	LED2	☑
PB5	n/a	Low	Output Pu...	No pull-up...	Very High	LED3	☑
PB6	n/a	Low	Output Pu...	No pull-up...	Very High	LED4	☑

- Timers kısmından TIM2 seçimi yapılır. Ardından Mod kısmından sadece Clock Source kısmını Internal Clock yaparız.



- Sistem clock 180MHz ayarlamıştık ve TIM2 clock veri yolu ise 45MHz'dir. fakat bunun 2 katı değeri alıyorlardı yani 90MHz'dir.





- Daha sonra Parameter Settings'den TIM2 1 saniye aralıklarla tekrarlı şekilde çalıştıracak değerler girilir. Sayma işlemi 0'dan başlamayıp 1'den başladığından 1 eksiğini alarak yazarız.
- Prescalaer bizim sayısının en üst seviyesini belirler. Burası 16 bit olduğundan en fazla 65535 yazabilirim. TIM2 clock hızı 90MHz olduğundan bunu kaça bölmeliyim diye soruyoruz. Bu clock hız için 45000'e bölüyoruz. Bu sayı da 44999 sayısı yapıyor.
- 90MHz'i 45000'e böldüğümüzde 2000 sayısı yapıyor. Bu değer Auto-reload oluyor. Bunun da 1 eksiğini yazıyoruz. Counter Period kısmında her seferinde taşma işlemi bittikten sonra tekrar bunu yükler. Yükleyeceği değeri yazarız.
- Sayma şeklini Up belirleyip yukarı doğru sayıyor.
- Auto-reload preload kısmında Enabled diyerek sayma bittiğinde başa dönmesini sağlarız.
- 2000 değeri yazmasaydık sonuç 2000 Hz olacağından sonucunda 0,0005s yani 1 sn=1000 ms olmak üzere 0,5 ms olacaktı.

Counter Settings

Prescaler (PSC - 16 bits value)	45000-1
Counter Mode	Up
Counter Period (AutoReload Register -.	2000-1
Internal Clock Division (CKD)	No Division
auto-reload preload	Enable

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler + 1)(Period + 1)}$$

$$UpdateEvent = \frac{90.000.000}{(45000)(2000)} = 1\,Hz = \frac{1}{1}s = 1\,s$$

- NVIC Settings kısmından TIM2 global interrupt Enabled yapılır. Bununla her güncellemede, sayıyı bitirmede bir interrupt oluşmasını sağlıyoruz.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM2 global interrupt	☑	0	0

Kod Kısmı

- hal_tim.c dosyasından Timer'ı Interrupt ile başlatmamız gerekiyor.

```
453 /**
454   * @brief  Starts the TIM Base generation in interrupt mode.
455   * @param  htim TIM Base handle
456   * @retval HAL status
457   */
458 HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)
```

```
 95   /* USER CODE BEGIN 2 */
 96   HAL_TIM_Base_Start_IT(&htim2);
 97   /* USER CODE END 2 */
```

- 96.satır çalışınca it.c dosyasına gider. Burada TIM için geçerli fonksiyonu 1sn'de bir çalıştırır.

```
202 /**
203   * @brief This function handles TIM2 global interrupt.
204   */
205 void TIM2_IRQHandler(void)
206 {
207   /* USER CODE BEGIN TIM2_IRQn 0 */
208
209   /* USER CODE END TIM2_IRQn 0 */
210   HAL_TIM_IRQHandler(&htim2);
211   /* USER CODE BEGIN TIM2_IRQn 1 */
212
213   /* USER CODE END TIM2_IRQn 1 */
214 }
```

- 210.satır ile bu fonksiyon dallanır ve dallanan fonksiyon içerisinde Callback fonksiyonu vardır. Bu fonksiyon sayesinde int main içerisinde de kodlarımı yazabilirim.

```
2038 /** @defgroup TIM_Exported_Functions_Group9 TIM Callbacks functions
2039   *  @brief   TIM Callbacks functions
2040   * @{
2041   */
2042 /* Callback in non blocking modes (Interrupt and DMA) *************************/
2043 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
```

- 1sn aralıklarla ledi yakıp söndürüyor.

```
57 /* Private user code ---------------------------------------------------------
58 /* USER CODE BEGIN 0 */
59 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
60 {
61     HAL_GPIO_TogglePin(GPIOB, LED1_Pin | LED2_Pin | LED3_Pin | LED4_Pin);
62 }
63 /* USER CODE END 0 */
```