External Interrupt

25 Aralık 2021 Cumartesi 00:54

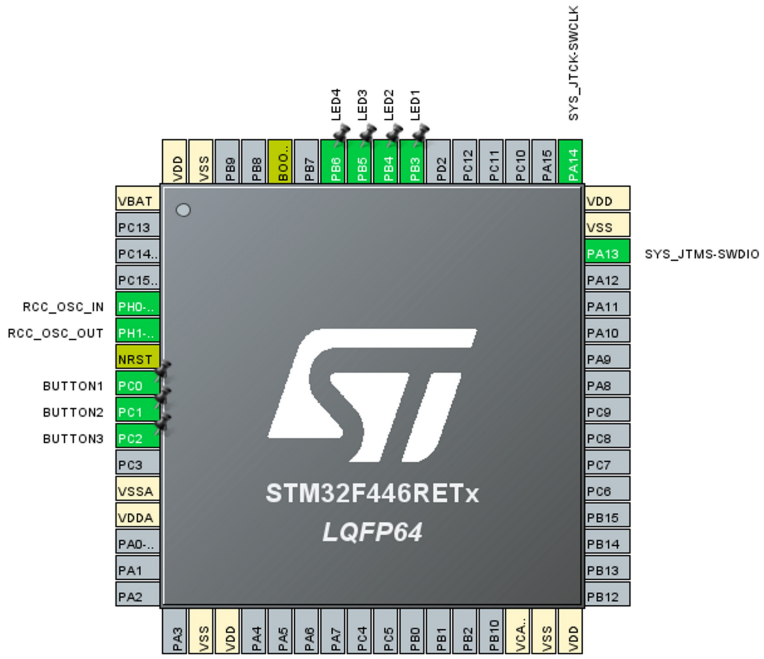
External Interrupt

- HAL

Teori

- Interrupt olduğunda gittiği yerdeki fonksiyon içerisindeki kod çalışır. Buradan çıktığı zaman kaldığı yere geri döner.
- https://www.youtube.com/watch?v=hikauabT0bM linkten örnek uygulamayı inceleyebiliriz.

Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-u...	Maximum ou...	User Label	Modified
PB3	n/a	Low	Output Push ...	No pull-up a...	Very High	LED1	☑
PB4	n/a	Low	Output Push ...	No pull-up a...	Very High	LED2	☑
PB5	n/a	Low	Output Push ...	No pull-up a...	Very High	LED3	☑
PB6	n/a	Low	Output Push ...	No pull-up a...	Very High	LED4	☑
PC0	n/a	n/a	External Inte...	Pull-down	n/a	BUTTON1	☑
PC1	n/a	n/a	External Inte...	Pull-down	n/a	BUTTON2	☑
PC2	n/a	n/a	External Inte...	Pull-down	n/a	BUTTON3	☑

- NVIC kısmından işaretlediğimiz pinlerin kesme işlemi yapabilmesi için Enabled kısımı işaretliyoruz.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line 0 interrupt	☑	0	0
EXTI line 1 interrupt	☑	0	0
EXTI line 2 interrupt	☑	0	0

Kod Kısmı

- İt.c dosyasına gittiğimizde 3 adet kesme fonksiyonları oluşturulmuştur.

```
205 void EXTI0_IRQHandler(void)
206 {
207   /* USER CODE BEGIN EXTI0_IRQn 0 */
208
209   /* USER CODE END EXTI0_IRQn 0 */
210   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
211   /* USER CODE BEGIN EXTI0_IRQn 1 */
212
213   /* USER CODE END EXTI0_IRQn 1 */
214 }
```

```c
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_1);
  /* USER CODE BEGIN EXTI1_IRQn 1 */

  /* USER CODE END EXTI1_IRQn 1 */
}

void EXTI2_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI2_IRQn 0 */

  /* USER CODE END EXTI2_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_2);
  /* USER CODE BEGIN EXTI2_IRQn 1 */

  /* USER CODE END EXTI2_IRQn 1 */
}
```

- İçerisinde yazılı olana Ctrl + Space tıkladığımızda hal_gpio.c dosyaya götürür. Burada en son satırda Callback fonksiyonunu çağırır.

```c
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
```

- Buna aynı şekilde gittiğimizde bu dosya üzerinde yer alır. Biz main.c dosyamızda bu fonksiyonu kullanıp kesme işlemi yapacağız.

```c
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(GPIO_Pin);
  /* NOTE: This function Should not be modified, when the callback is needed,
           the HAL_GPIO_EXTI_Callback could be implemented in the user file
   */
}
```

- While döngüsü ile tüm pinleri yaktık ve count değişkeni değerinde delay süresi verdik.

```c
/* Private includes ----------
/* USER CODE BEGIN Includes */
int count=0 , i=0;
/* USER CODE END Includes */
```

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

    /* USER CODE BEGIN 3 */
      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_All);
      HAL_Delay(count);
  }
  /* USER CODE END 3 */
}
```

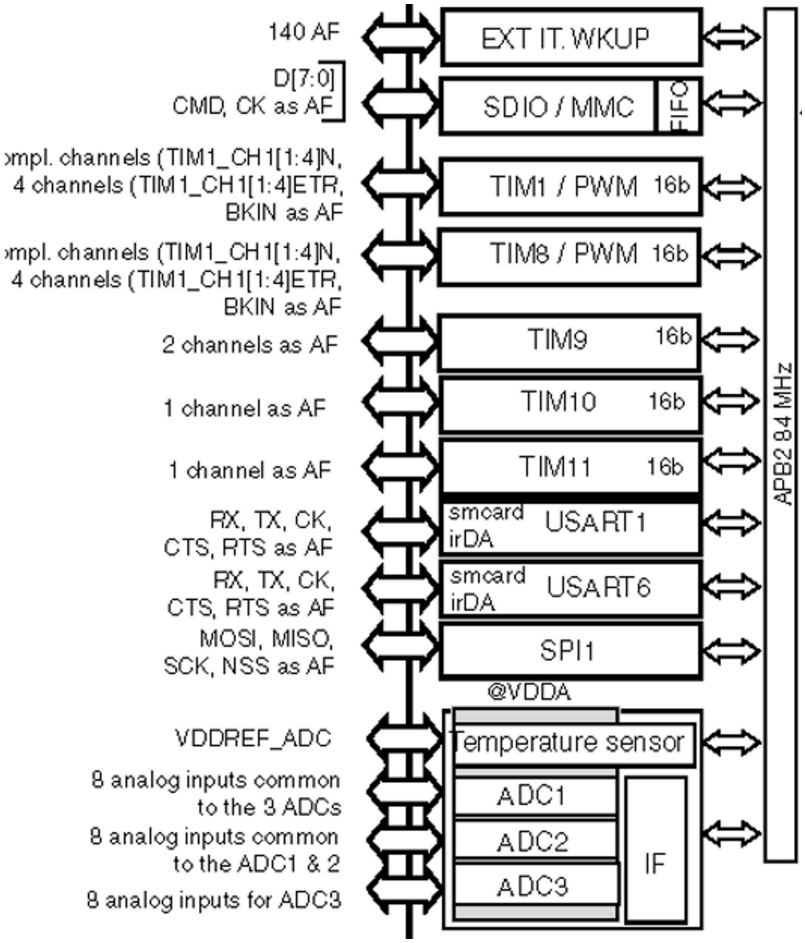
- Count değişkenini kesme olduğunda her buton için ayrı delay süreleri verdik.

```c
/* Private user code ---------------------------
/* USER CODE BEGIN 0 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    i++;
    if(HAL_GPIO_ReadPin(GPIOC, BUTTON1_Pin))
    {
        if(i==1)
        {
            count=1000;
        }
        else if(i==2)
        {
            count=750;
        }
        else if(i==3)
        {
            count=500;
        }
        else
        {
            count=250;
            i=0;
        }
    }
    else if(HAL_GPIO_ReadPin(GPIOC, BUTTON2_Pin))
    {
        count=100;
    }
    else
    {
        count=50;
    }
}
/* USER CODE END 0 */
```

➢ REGISTER

Konfigürasyon Kısmı

- External Interrup'ın clock hattı APB2'ye gidiyor.



RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 EN	TIM10 EN	TIM9 EN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	Reser-ved	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART 6 EN	USART 1 EN	Reserved		TIM8 EN	TIM1 EN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

Bit 14 **SYSCFGEN:** System configuration controller clock enable
Set and cleared by software.
0: System configuration controller clock disabled
1: System configuration controller clock enabled

- 14.pini 1 yapıyoruz.

```
RCC->APB2ENR |= 0x00004000;            //SYSCGFEN
```

- 3 tane interrupt pini kullanacağımızdan öncelikle bunları aktif edip, öncelik sırası belirlemeliyiz.
- core_cm4.h kütüphanesinde bu fonksiyonları görebiliriz.

```
1452 __STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
```

- İkinci değere öncelik numarası yazıyoruz. Ne kadar küçük olursa öncelik sıralamasında önde olur.

```
1535 __STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

```
NVIC_EnableIRQ(EXTI0_IRQn);
NVIC_EnableIRQ(EXTI1_IRQn);
NVIC_EnableIRQ(EXTI2_IRQn);
```

```
NVIC_EnableIRQ(EXTI0_IRQn);
NVIC_EnableIRQ(EXTI1_IRQn);
NVIC_EnableIRQ(EXTI2_IRQn);

NVIC_SetPriority(EXTI0_IRQn, 0);
NVIC_SetPriority(EXTI0_IRQn, 1);
NVIC_SetPriority(EXTI0_IRQn, 2);
```

- Interrupt olarak mı event olarak mı kullanacağımızı belirliyoruz. Interrupt olarak kullanacağız.

Interrupt mask register (EXTI_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **MRx:** Interrupt mask on line x

0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

- İlk 3 pine 1 yazıyoruz.

```
EXTI->EMR |= x00000007;
```

- Alçalan kenar mı yükselen kenar mı kullanacağımızı belirliyoruz. Yükselen kenar kullanacağız.

Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx:** Rising trigger event configuration bit of line x

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line

İlk 3 pine 1 yazıyoruz.

```
EXTI->RTSR |= x00000007;
```

- Butonları external interrupt için aktif etmemiz gerekiyor.

Buton için A portunu kullanıyoruz.

SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)
These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin

```
SYSCFG->EXTICR[0] = 0x00000000;
```

- RCC, GPIO ve EXTI için yazdığımız fonksiyonlar aşağıdaki gibidir.

```
8  void RCC_Config(void)
9  {
10     RCC->CR |= 0x00030000;               //HSEON, HSERDY
11     while(!(RCC->CR & 0x00020000));      //HSERDY
12     RCC->CR |= 0x00080000;               //CSSON
13     RCC->CFGR = 0x00000000;
14     RCC->PLLCFGR |= 0x00400000;          //PLLSRC
15     RCC->PLLCFGR |= 0x00000004;          //PLLM 4
16     RCC->PLLCFGR |= 0x00002A00;          //PLLN 168
17     RCC->PLLCFGR |= 0x00000000;          //PLLP 2
18     RCC->CR |= 0x01000000;               //PLLON
19     while(!(RCC->CR & 0x02000000));      //PLLRDY
20     RCC->CFGR |= 0x00000001;             //SW
21     while(!(RCC->CR & 0x00000001));      //SWS
22 }
```

```
24 void GPIO_Config(void)
25 {
26     RCC->AHB1ENR |= 0x00000009;          //A, D clock enable
27
28     GPIOD->MODER |= 0x55000000;         //PD12, PD13, PD14, PD15
29     GPIOD->OTYPER |= 0x00000000;        //Output push-pull
30     GPIOD->OSPEEDR |= 0xFF000000;       //Very high speed
31     GPIOD->PUPDR |= 0x00000000;         //No pull-up, pull-down
32 }
```

```
34 void EXTI_Config(void)
35 {
36     RCC->APB2ENR |= 0x00004000;          //SYSCGFEN
37
38     NVIC_EnableIRQ(EXTI0_IRQn);
39     NVIC_EnableIRQ(EXTI1_IRQn);
40     NVIC_EnableIRQ(EXTI2_IRQn);
41
42     NVIC_SetPriority(EXTI0_IRQn, 0);
43     NVIC_SetPriority(EXTI0_IRQn, 1);
44     NVIC_SetPriority(EXTI0_IRQn, 2);
45
46     EXTI->EMR |= 0x00000007;
47     EXTI->RTSR |= 0x00000007;
48
49     SYSCFG->EXTICR[0] = 0x00000000;
50 }
```

Kod Kısmı

- İlk önce while döngüsünde tüm ledleri açıyoruz.

```c
int main(void)
{
    RCC_Config();
    SystemCoreClockUpdate();
    GPIO_Config();
    EXTI_Config();

  while (1)
  {
      GPIOD->ODR |= 0x0000F000;
  }
}
```

- Pinde kesme olup olmadığını öğrenmemiz gerekiyor. Bitte değer olduğunda kesmeye girmiş oluyor. Tekrar 1 yazarak bayrağı indiriyoruz.

Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **PRx:** Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line.
This bit is cleared by programming it to '1'.

- Eğer kesmeye giriliyorsa 2 saniye boyunca belirlediğimiz ledi açıp diğerlerini kapatıyoruz.

```c
void EXT0_IRQHandler()
{
    if(EXTI->PR & 0x00000001)
    {
        GPIOD->ODR |= 0x00001000;
        delay(33600000);
    }
    EXTI->PR = 0x00000001;
}
```

```c
void EXT1_IRQHandler()
{
    if(EXTI->PR & 0x00000002)
    {
        GPIOD->ODR |= 0x00002000;
        delay(33600000);
    }
    EXTI->PR = 0x00000002;
}
```

```c
void EXT2_IRQHandler()
{
    if(EXTI->PR & 0x00000004)
    {
        GPIOD->ODR |= 0x00004000;
        delay(33600000);
    }
    EXTI->PR = 0x00000004;
}
```