

STM32F103 Programlama

23 Şubat 2021 Salı 00:18

- <https://www.youtube.com/playlist?list=PL-p1dQWEJtWbHbnawz64RFk0JwpOSS4jD>
- <https://www.youtube.com/playlist?list=PLre4S1H8v3bQRzjCMNY1BoAihAKKgVWQ>

- ☒ [01 Dahili Led Yakma](#)
- ☒ [02 Harici Led Yakma](#)
- ☒ [03 Buton ile Led Yakma](#)
- ☒ [04 Timer Interrupt](#)
- ☒ [05 External Interrupt](#)
- ☒ [06 Tek Kanal ADC](#)
- ☒ [07 ADC DMA](#)
- ☒ [08 Çok Kanal ADC](#)
- ☒ [09 ADC Interrupt](#)
- ☒ [10 PWM](#)
- ☒ [11 USART](#)
- ☒ [12 I2C EEPROM](#)

01 Dahili Led Yakma

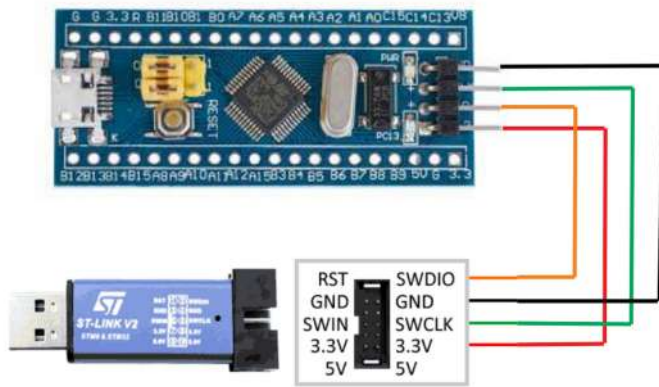
23 Şubat 2021 Salı 01:07

- <https://youtu.be/-d6WD1UBiPI>
- <https://youtu.be/bJhIfAfpWC0>
- <https://youtu.be/d9XVKVo50oc>

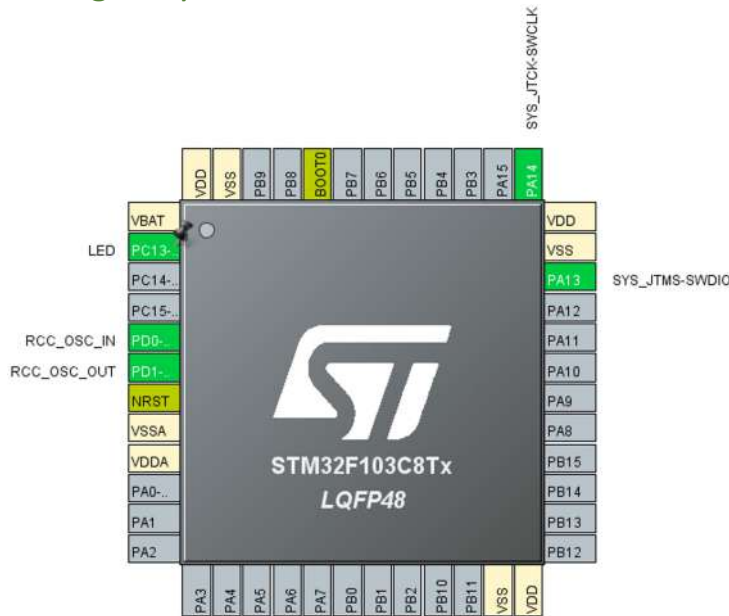
01 Dahili Led Yakma

Teorik Kısım

- Kodun nasıl yazıldığını bilmeyebiliriz. Bunun için Keil'de Functions kısmından CubeIDE'de Core ve Drivers dosya kısmından bakabiliriz. Burdaki .h dosyalarında fonksiyonların tanımlamaları olur. .c dosyasında kullanabileceğimiz fonksiyonlar gözükür. Nasıl kullanıldığını öğrenmek için çift tıklarız.



Konfigürasyon Kısım



- System Core kısmından SYS tıklıyoruz ve Debug'da Serial Wire diyoruz. Çünkü St-Link kullanabilmek için bu işlemi yapmamız gerekiyor.

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-...	Maximum out...	User Label	Modified
PA13	SYS_JTMS-SWDIO	n/a	n/a	n/a	n/a		<input type="checkbox"/>
PA14	SYS_JTCK-SWCLK	n/a	n/a	n/a	n/a		<input type="checkbox"/>

- RCC tıklıyoruz HSE'de Crystal/Ceramic Resonator işaretliyoruz.


```

48 /* Private function prototypes -
49 void SystemClock_Config(void);
50 static void MX_GPIO_Init(void);

```

- Kod aslında int main(void)'den başlar while'ın sonunda biter.

```

64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99     }

```

- CubeMX kısmında yaptığımız sistem clock konfigürasyonunda yaptıklarımız kod içerisine kendi dahil olur.

- GPIO başlatma fonksiyonlarını barındırır.
- Bu fonksiyona hal_gpio.c dosyasından ulaşırız.

```

465 void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)

```

- Write pin komutun üç parametresi sırasıyla GPIO tipi, pini, durumudur.
- PC13 pini için SET ile 0, RESET ile 1 yapılır.

```

94 while (1)
95 {
96     /* USER CODE END WHILE */
97
98     /* USER CODE BEGIN 3 */
99     HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
100     HAL_Delay(1000);
101     HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
102     HAL_Delay(1000);
103
104 }

```

- Kod yükleme işlemi yaparken önce hatamız var mı öğrenmek için önce Build yapılır.
- Keil'de Options and Target kısmından ayarlamalar yaparız. Output'tan Hex kodunu

oluşturabiliriz. Debug kısmından St-Link Debugger seçilir ve Setting kısmından ayarlamalar yapılır ardından load yapılır.



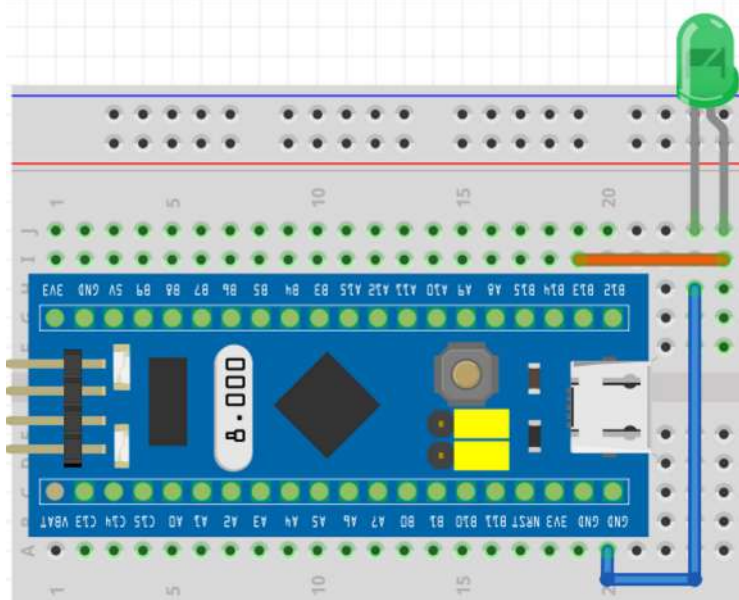
- Kodu St-Link Utility programı ile yüklemek istersek önce aygıta bağlanmak için Connect to the Target diyoruz.
Hex kodu yüklemek için Program Verify tıklıyoruz. MDK-ARM içinde dosya isimli olana tıklıyoruz ve karşımıza hex dosyası gözükmüş oluyor.
- CubeIDE'de Run Configurations kısmından Debbugger tıklanır ve Debug Probe kısmından ST-LINK (OpenOCD) seçilir. Burda Show generator options tıklanır Hide generator options yapılır ayrıca Reset Mode Software system reset seçilir ve Apply deyip OK denir.

23 Şubat 2021 Salı 01:07

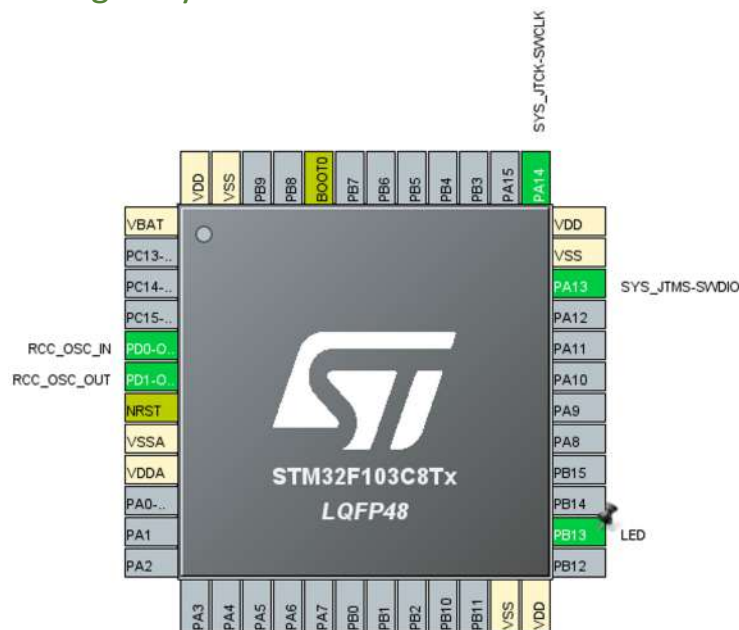
- https://youtu.be/s5_FLd9RKVA

02 Harici Led Yakma

Devre Kısmı



Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PB13	n/a	Low	Output Push...	No pull-up an...	Low	LED	<input checked="" type="checkbox"/>

Kod Kısmı

- Toggle 0 olan durumu 1, 1 olan durumu 0 yapar.
- Port yazarken pine verdiğimiz isim şekli olan "LED_GPIO_Port" yerine direk portu "GPIOB" olarak yazabiliriz. Bu tanımlamayı main.h metninden bakarak yazıyoruz. Aynı şekilde aynı yerden pin adını da "LED_Pin" yerine "GPIO_PIN_13" şeklinde yazabiliriz.
- Bu tanımlamalara üzerine gelip Keil'de F12 ile CubeIDE'de F3 ile bakabiliriz. Aşağıda

F3'e tıkladığımızda main.h kütüphanesindeki bu satırları gösteriyor.

```
#define LED_Pin GPIO_PIN_13
```

```
#define LED_GPIO_Port GPIOB
```

- Bu fonksiyona hal_gpio.c dosyasından ulaşırız.

```
487 void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
```

```
94 while (1)
95 {
96     /* USER CODE END WHILE */
97
98     /* USER CODE BEGIN 3 */
99     HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
100     HAL_Delay(200);
101
102 }
```

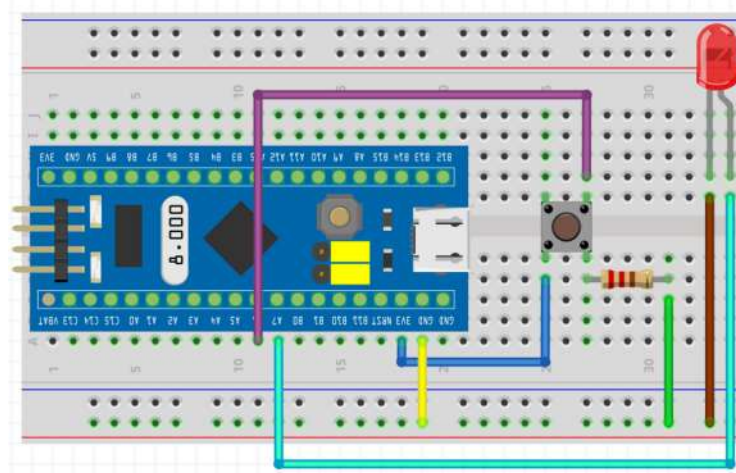

03 Buton ile Led Yakma

27 Şubat 2021 Cumartesi 18:36

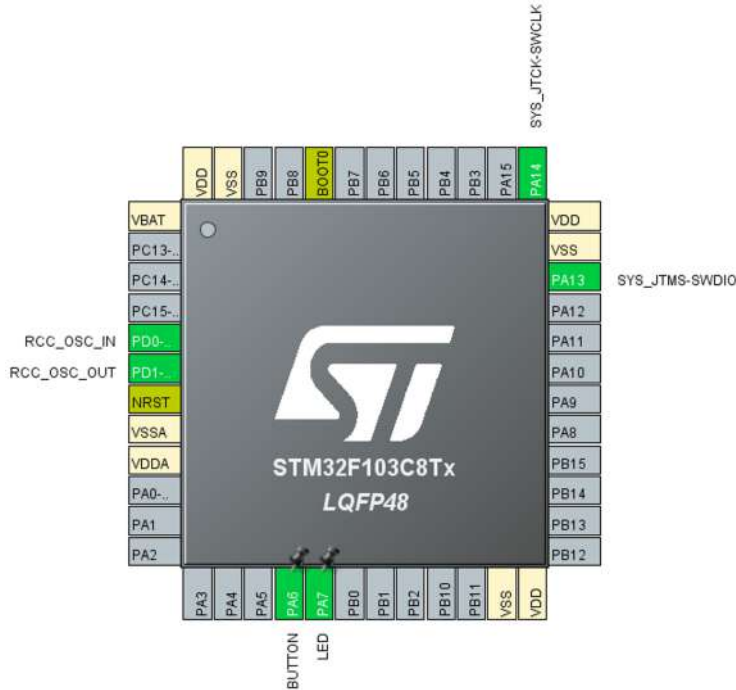
- <https://youtu.be/XimVOa4tQGf>

03 Buton ile Led Yakma

Devre Kısım



Konfigürasyon Kısım



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA6	n/a	n/a	Input mode	Pull-down	n/a	BUTTON	✓
PA7	n/a	Low	Output Push...	No pull-up an...	Low	LED	✓

Kod Kısım

- Bu fonksiyona hal_gpio.c dosyasından ulaşırız.

```
431 GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
```

```
94  while (1)
95  {
96      /* USER CODE END WHILE */
97
98      /* USER CODE BEGIN 3 */
99      if(HAL_GPIO_ReadPin(GPIOA,BUTTON_Pin)==1)
100      {
101          HAL_GPIO_WritePin(GPIOA,LED_Pin,GPIO_PIN_SET);
102      }
103      else
104      {
105          HAL_GPIO_WritePin(GPIOA,LED_Pin,GPIO_PIN_RESET);
106      }
107
108  }
```

04 Timer Interrupt

27 Şubat 2021 Cumartesi 00:03

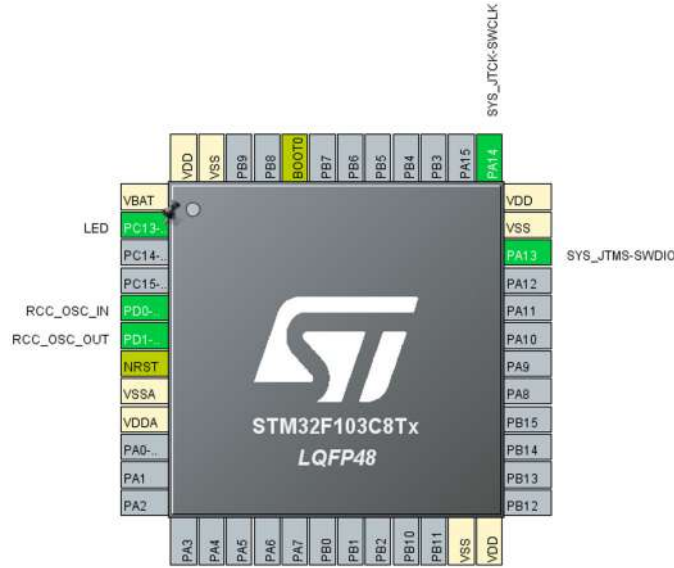
- https://youtu.be/-FO_jc_iag
- <https://youtu.be/KLtSiVqjKkk>

04 Timer Interrupt

Teori Kısmı

- Timer saymaya başlar belirttiğimiz sayıdan sonra taşma işlemi oluyor ve başa dönüyor. Limite göre de istediğimiz işlemi yapıyoruz. Tasma işlemine kesme adını veriyoruz.
- Timer, delaydan farkı arka tarafta çalışmasıdır.

Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up/P...	Maximum ou...	User Label	Modified
PC13-TAMP...	n/a	Low	Output Push...	No pull-up and ...	Low	LED	✓

- Timer1 için ayarlama yapıyoruz. Timer için girilen değerlere göre saniye hesabını aşağıdaki formül ile yaparız. Bizim timer 1 saniye aralıklarla tekrarlı şekilde çalışır.
- Timers kısmında TIM1 tıklarız ve öncelikle Mode kısmında Clock Source'dan Internal Clock seçilir. Bu seçim bizim dahili olanı seçtiğimizi gösterir. ETR2 ise harici olanı kullanacağımız anlamına gelir.
- Daha sonra Parameter Settings'den aşağıdaki değerler girilir. Prescaler bizim sayısının en üst seviyesini belirler. Sayma işlemi 0'dan başlamayıp 1'den başladığından 1 eksiğini alarak yazarız. Sayma şeklini Up belirleyip yukarı doğru sayıyor. Counter Period kısmında her seferinde taşma işlemi bittikten sonra tekrar bunu yükler. Yükleyeceği değeri yazarız. Auto-reload preload kısmında Enable diyerek sayma bittiğinde başa dönmesini sağlarız.

Counter Settings

Prescaler (PSC - 16 bits value)	36000-1
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits val...	2000-1
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable

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

$$UpdateEvent = \frac{72.000.000}{(36000)(2000)} = 1 Hz = \frac{1}{1} s = 1 s$$

- NVIC Settings kısmından TIM1 update interrupt Enabled yapılır. Bununla her güncellemede, sayıyı bitirmede bir interrupt oluşmasını sağlıyoruz.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM1 break interrupt	<input type="checkbox"/>	0	0
TIM1 update interrupt	<input checked="" type="checkbox"/>	0	0
TIM1 trigger and commutation interrupts	<input type="checkbox"/>	0	0
TIM1 capture compare interrupt	<input type="checkbox"/>	0	0

Kod Kısmı

- 92.satırdaki kod ile interrupt modunda Timer'ı çalıştırır.
- Bu fonksiyona hal_tim.c dosyasına baktığımızda htim önünde yıldız görürüz. Bu önüne "&" işareti kullanılır.

```
458 HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)
```

```
88  /* Initialize all configured peripherals */
89  MX_GPIO_Init();
90  MX_TIM1_Init();
91  /* USER CODE BEGIN 2 */
92  HAL_TIM_Base_Start_IT(&htim1);
93
94  /* USER CODE END 2 */
```

- Az önce yazılan kod ile it.c dosyasında fonksiyon altına otomatik gelir. Gelmez ise timer'ın interrupt kısmı başlamaz.

```
205 void TIM1_UP_IRQHandler(void)
206 {
207     /* USER CODE BEGIN TIM1_UP_IRQn 0 */
208
209     /* USER CODE END TIM1_UP_IRQn 0 */
210     HAL_TIM_IRQHandler(&htim1);
211     /* USER CODE BEGIN TIM1_UP_IRQn 1 */
212
213     /* USER CODE END TIM1_UP_IRQn 1 */
214 }
```

- Timer1 için yapması gereken işlemi yazıyoruz. Öncesinde değişkeni yazıyoruz.

```
45 /* USER CODE BEGIN PV */
46 uint16_t sayi=0;
47 /* USER CODE END PV */
```

- hal_tim.c dosyasından HAL_TIM_PeriodElapsedCallback fonksiyonunu çekeriz. Timer'ın yaptığı işlemler buradan çağırılır.

```
5511 __weak void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
```

- Hangi timer'ın yapısını kullanıyorsam önce onu gösteriyorum. 224.satırda eğer taşma yapan timer TIM1 ise sayı değişkenini 1 arttır diyorum. Yani taşma işlemi olursa istediğimiz işlemleri yapabiliyoruz.

- Sadece 226.satırdaki kod çalışsaydı 1 saniyelik gecikme yapıp ledimiz yanıp sönecekti. Fakat 1 saniye'de bir sayı değişkenini 1 arttırıp 5 olduğu zaman ledi söndürüyor.

```
222 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
223 {
224     if(htim->Instance==TIM1)
225     {
226         //HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
227         sayi++;
228     }
229 }
```

```
98  while (1)
99  {
100     /* USER CODE END WHILE */
101
102     /* USER CODE BEGIN 3 */
103         if(sayi==5)
104         {
105             HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
106         }
107     }
108     /* USER CODE END 3 */
109 }
```

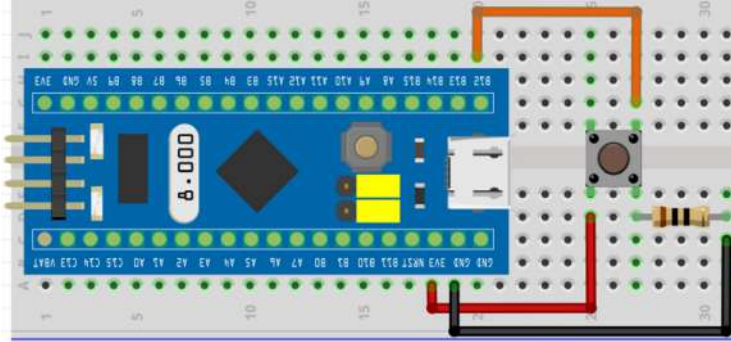
05 External Interrupt

1 Mart 2021 Pazartesi 19:18

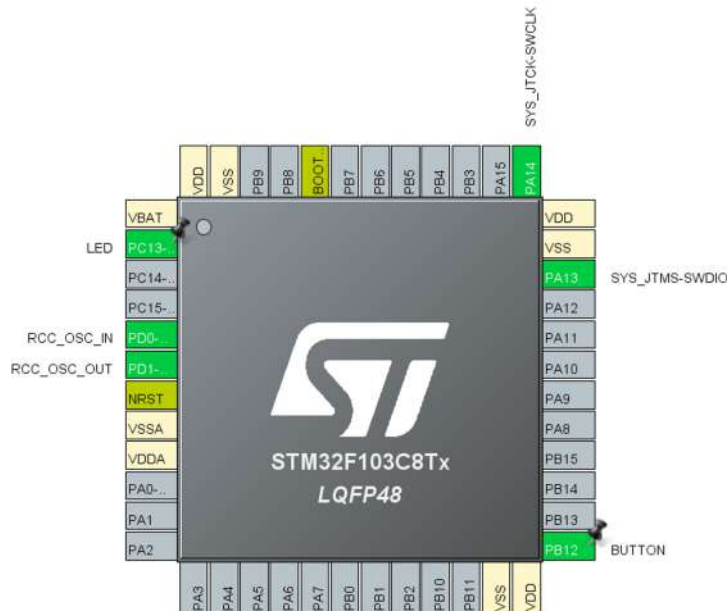
- <https://youtu.be/QN16t6J1VKA>
- <https://youtu.be/hikauabT0bM>

05 External Interrupt

Devre Kısımı



Konfigürasyon Kısımı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PB12	n/a	n/a	External Inte...	Pull-down	n/a	BUTTON	✓
PC13-TAMP...	n/a	Low	Output Push...	No pull-up an...	Low	LED	✓

- PB12 pinini buton adına GPIO_EXTI12 olarak ayarlıyoruz.
- Bunu yaptıktan sonra kesme ayarı yapmamız gerekiyor. NVIC kısmından ilgili satır olan en alttakini işaretliyoruz ve Preemption Priority değerini 15 yapıyoruz. Bunun anlamı öncelik sırasını belirlemek için kullanıyoruz ve biz en son değeri seçiyoruz.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	15	0

- NVIC birimi mikrodenetleyicideki interrupt işlemlerini kontrol eden birimdir.

Kod Kısmı

- İlk önce söndürme işlemi yapıyoruz.

```
88  /* USER CODE BEGIN 2 */
89  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);
90  /* USER CODE END 2 */
```

```
44  /* USER CODE BEGIN PV */
45  uint16_t sayi=0;
46  /* USER CODE END PV */
```

- Kesme işlemi olduğunda nasıl davranacağını aşağıdaki gibi belirleriz.
- Bu fonksiyona hal_gpio.c dosyasından ulaşırız.

```
561__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

```
178 /* USER CODE BEGIN 4 */
179 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
180 {
181     HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
182     sayi++;
183 }
184
185 /* USER CODE END 4 */
```

- Interrupt için Callback fonksiyonlarını kullandık ayrıca Handler fonksiyonlarını da kullanabiliydik fakat bu fonksiyon Callback fonksiyonlarını arkada tarafta çağırıyor.
- Handler fonksiyonuyla yaparsak it.c fonksiyonunda kesme olduğunda EXTI15_10_IRQHandler fonksiyonu altında yazdığımız kodlar çalışmış olacak.

```
205 void EXTI15_10_IRQHandler(void)
206 {
207     /* USER CODE BEGIN EXTI15_10_IRQn 0 */
208     HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
209     while (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_12))
210
211     /* USER CODE END EXTI15_10_IRQn 0 */
212     HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12);
213     /* USER CODE BEGIN EXTI15_10_IRQn 1 */
```



```

205 void EXTI15_10_IRQHandler(void)
206 {
207     /* USER CODE BEGIN EXTI15_10_IRQn 0 */
208     HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
209     while (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_12))
210
211     /* USER CODE END EXTI15_10_IRQn 0 */
212     HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12);
213     /* USER CODE BEGIN EXTI15_10_IRQn 1 */
214
215     /* USER CODE END EXTI15_10_IRQn 1 */
216 }

```

- Kod çalıştığında led yanıyor daha sonra butona bastığımızda led sönüyor.

1 Mart 2021 Pazartesi 23:26

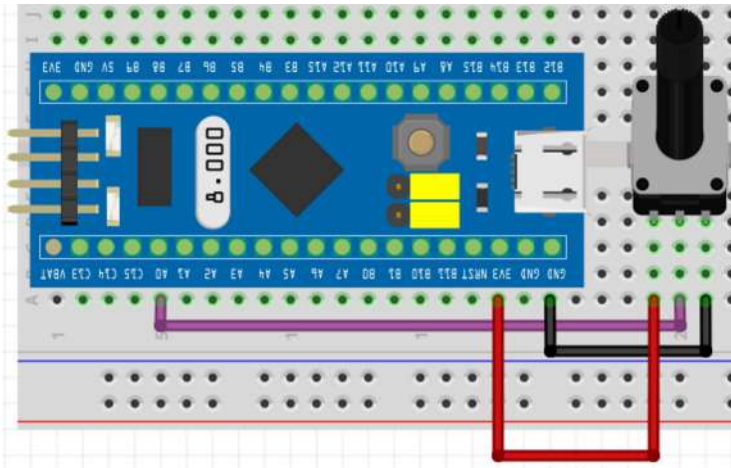
- <https://youtu.be/3uDI-H8o7Xg>
- <https://youtu.be/hsMzpDI1hx0>

06 Tek Kanal ADC

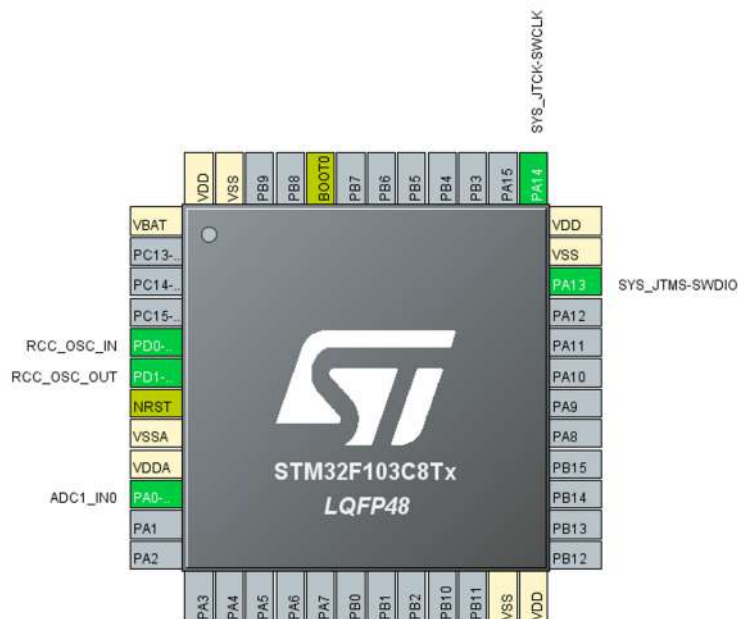
Teori Kismi

- Mikrodenetleyicimizde 12 bitlik ADC yapısı var.

Devre Kismi



Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	ADC1_IN0	n/a	Analog mode	n/a	n/a		<input type="checkbox"/>

- Analog değer okumak için potansiyometreyi PA0'a bağladık. Bunun için Analog kısmından ADC1'den IN0 tıklarız.
- Daha sonra Parameter Settings'den Continuous Conversion Mode Enabled yapılır. Sürekli çevrim modu anlamına gelir ve tekrar tekrar okuma durumu yapar eğer çalıştırmazsak bir kere okur daha sonra değer okumaz.

ADC_Settings

Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled

Kod Kısmı

- "&" işaretinden sonra yazdığımız bizim kaynağımız oluyor.
- Önce ADC işlemini başlattık.

```
88  /* Initialize all configured peripherals */
89  MX_GPIO_Init();
90  MX_ADC1_Init();
91  /* USER CODE BEGIN 2 */
92  HAL_ADC_Start(&hadc1);
93  /* USER CODE END 2 */

45  /* USER CODE BEGIN PV */
46  uint16_t adc_deger=0;
47  /* USER CODE END PV */

97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      adc_deger=HAL_ADC_GetValue(&hadc1);
103  }
104  /* USER CODE END 3 */
105 }
```

- $2^{12}=4096$ olduğundan en fazla bu değeri gösterebilir.

07 ADC DMA

1 Mart 2021 Pazartesi 23:26

- <https://youtu.be/tsZrPbWYKfI>

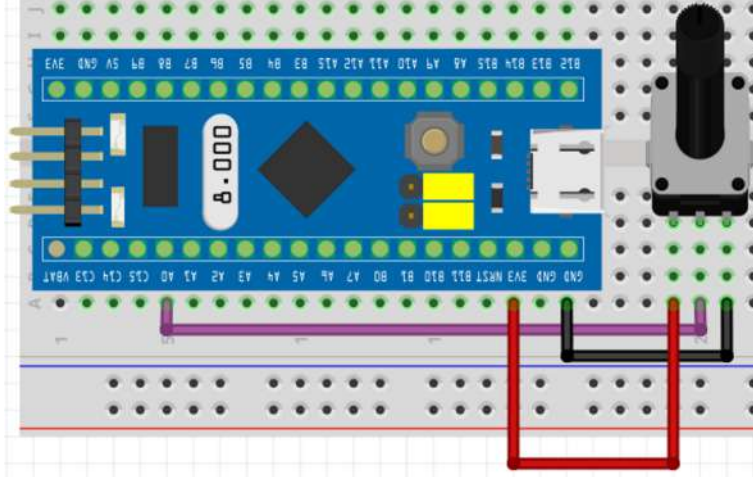
07 ADC DMA

Teori Kısmı

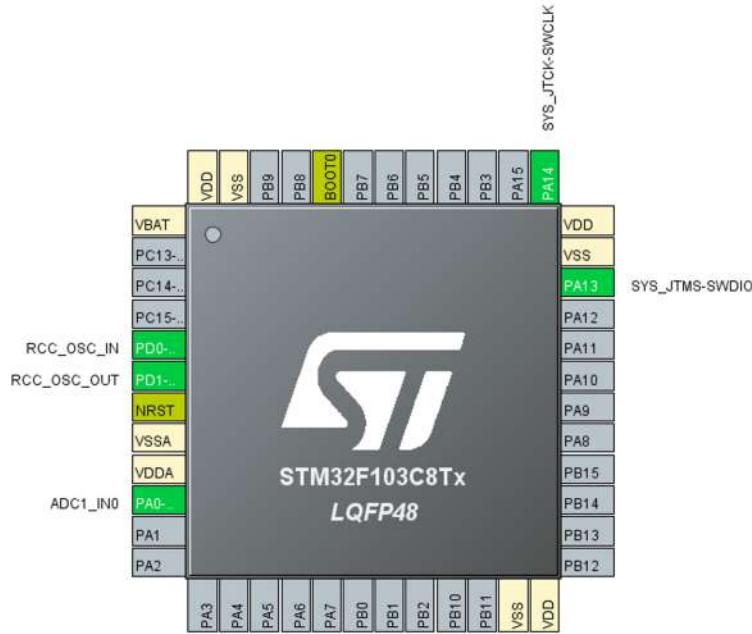
- İşlemci yaptığı işlemleri bellekte saklarlar. O bellekteki bilgiyi tekrar isteyerek tekrardan işleriz.
- DMA yönetimi ile ilgili tablo bize hangi kanalın hangi DMA'da olduğunu gösteriyor.

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC	ADC	ADC	-	-	-
SPI	-	SPI1_RX SPI1_TX	SPI2_RX SPI2_TX	-	-
USART	-	USART1_TX USART3_TX	USART1_RX USART3_RX	USART1_TX USART2_TX	USART1_RX USART2_RX
I2C	-	I2C1_TX I2C1_RX	I2C2_TX I2C2_RX	-	-
TIM1	-	TIM1_CH1 TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	-	TIM1_CH3 TIM1_UP
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	TIM3_CH1 TIM3_TRIG	-
TIM6	-	-	TIM6_UP	-	-
TIM7	-	-	-	TIM7_UP	-
TIM15	-	-	-	-	TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM
TIM16	-	-	TIM16_CH1 TIM16_UP	TIM16_CH1 TIM16_UP	-
TIM17	TIM17_CH1 TIM17_UP	TIM17_CH1 TIM17_UP	-	-	-

Devre Kısmı



Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	ADC1_IN0	n/a	Analog mode	n/a	n/a		<input type="checkbox"/>

- Önceki uygulamamızdan ekstra olarak ADC1 ayarları kısmından DMA Settings tıklayıp Add işleminden ADC1 seçilir ve burdan Mode kısmı sürekli olan Circular ile Memory kısmındaki Word seçilir.

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

Add

Delete

DMA Request Settings

Mode	Circular	Increment Address	<input type="checkbox"/>	Peripheral	<input type="checkbox"/>	Memory	<input checked="" type="checkbox"/>
Data Width	Half Word					Word	<input type="checkbox"/>

Kod Kısmı

- Değişkenimizi dizi olarak tanımladık. Çünkü, bizden dizi olarak istenilecek.

```

37 /* Private macro -----
38 /* USER CODE BEGIN PM */
39 uint32_t adc_deger[1];
40 /* USER CODE END PM */

```

- ADC'yi başlatırken DMA kullandığımızdan buna göre yazıyoruz. Bunu yazarken bizden ekstra iki parametre istiyor. Birincisi dizi halindeki değişken adı, ikincisi dizinin uzunluğu yani kaç data alacaksın, kaç diziye kaydedeceksin anlamındadır. Biz burada bir kanal kullandığımızdan bir yazdık.

```

90 /* Initialize all configured peripherals */
91 MX_GPIO_Init();
92 MX_DMA_Init();
93 MX_ADC1_Init();
94 /* USER CODE BEGIN 2 */
95 HAL_ADC_Start_DMA(&hadc1,adc_deger,1);
96 /* USER CODE END 2 */

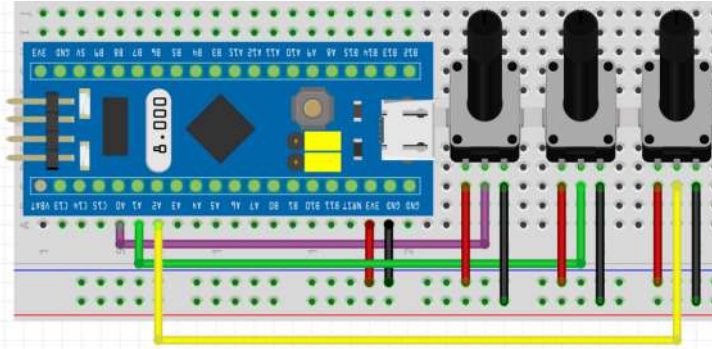
```


5 Mart 2021 Cuma 02:36

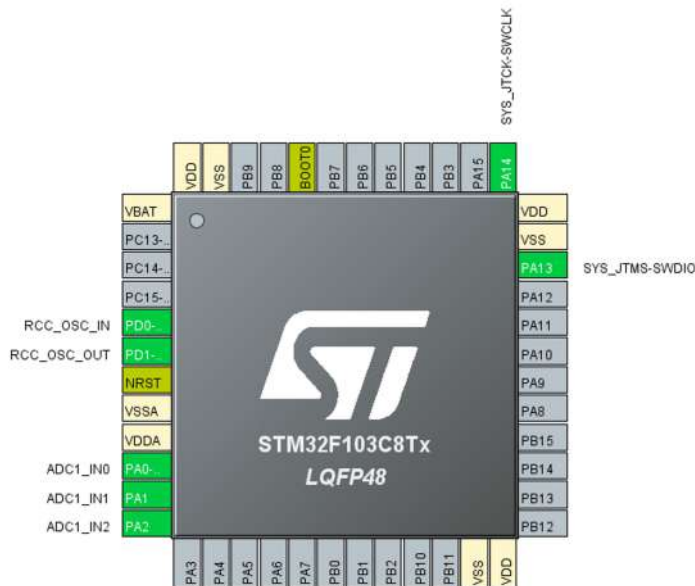
- <https://youtu.be/tsZrPbWYKfl>
- https://youtu.be/oTO_kRq2SmA

08 Çok Kanal ADC

Devre Kısımı



Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	ADC1_IN0	n/a	Analog mode	n/a	n/a		<input type="checkbox"/>
PA1	ADC1_IN1	n/a	Analog mode	n/a	n/a		<input type="checkbox"/>
PA2	ADC1_IN2	n/a	Analog mode	n/a	n/a		<input type="checkbox"/>

- Number of Conversion kısmında toplam kanal sayısını giriyoruz. Biz IN0, IN1 ve IN2 seçtiğimiz için 3 adet olarak giriyoruz ve Scan Conversion Mode otomatik Enabled oluyor. Bu da tarama işlemini gerçekleştiriyor.
- Rank ile kanalları seçebiliyoruz. Tekrarlanma zamanlarını belirleyebiliyoruz.
- Tekrardan DMA ayarlarını yapıyoruz.

ADC_Settings

Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled

ADC_Regular_ConversionMode

Enable Regular Conversions	Enable
Number Of Conversion	3
External Trigger Conversion Source	Regular Conversion launched by software
Rank	1
Rank	2
Rank	3

Rank	1
Channel	Channel 0
Sampling Time	1.5 Cycles
Rank	2
Channel	Channel 1
Sampling Time	1.5 Cycles
Rank	3
Channel	Channel 2
Sampling Time	1.5 Cycles

Kod Kısmı

- Toplam üç kanal olduğundan 3 yazdık.

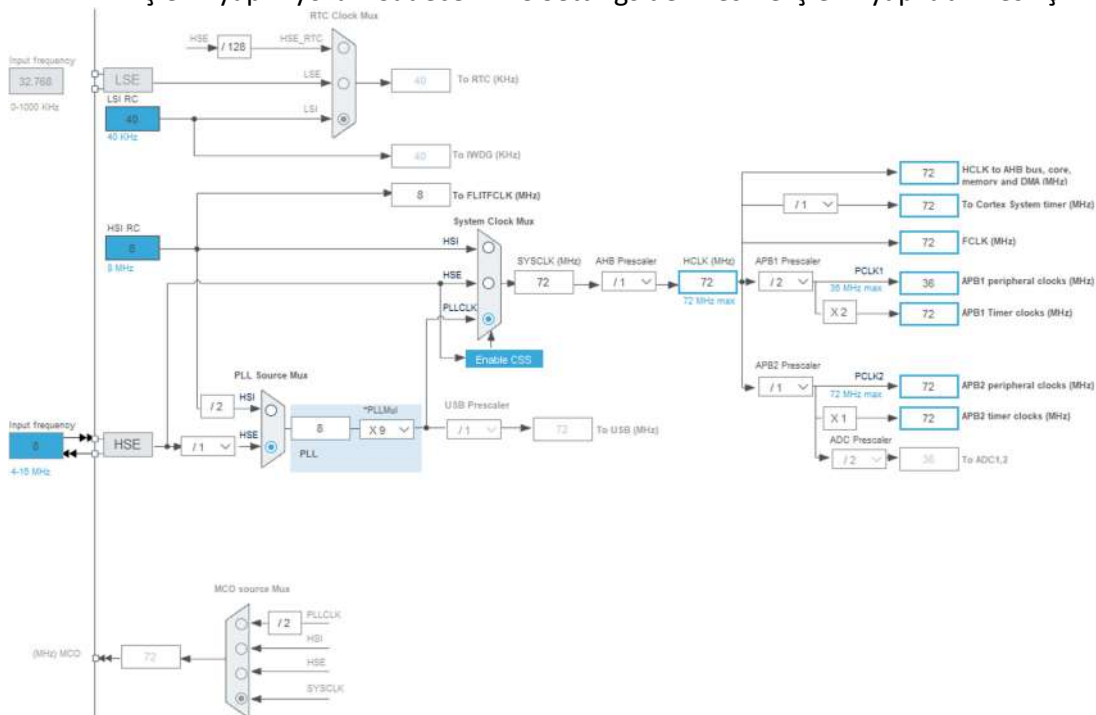
```
59 /* Private user code ---
60 /* USER CODE BEGIN 0 */
61 uint32_t adc_deger[3];
62 /* USER CODE END 0 */

90 /* Initialize all configured peripherals */
91 MX_GPIO_Init();
92 MX_DMA_Init();
93 MX_ADC1_Init();
94 /* USER CODE BEGIN 2 */
95 HAL_ADC_Start_DMA(&hadc1,adc_deger,3);
```


1 Mart 2021 Pazartesi 23:26

Teori Kismi

- ## Devre Kismi



Kod Kısmı

```

37  /* Private macro -----
38  /* USER CODE BEGIN PM */
39  uint32_t adc_deger;
40  /* USER CODE END PM */

```

```

88  /* Initialize all configured peripherals */
89  MX_GPIO_Init();
90  MX_ADC1_Init();
91  /* USER CODE BEGIN 2 */
92  HAL_ADC_Start_IT(&hadc1);
93  /* USER CODE END 2 */

```

- Interrupt olursa bu bölüme gelip buradaki kodları çalıştırır. Eğer ADC1 ise değeri değişkene yazdırır.

```

210  /* USER CODE BEGIN 4 */
211  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
212  {
213      if(hadc->Instance==ADC1)
214      {
215          adc_deger=HAL_ADC_GetValue(&hadc1);
216      }
217  }
218  /* USER CODE END 4 */

```

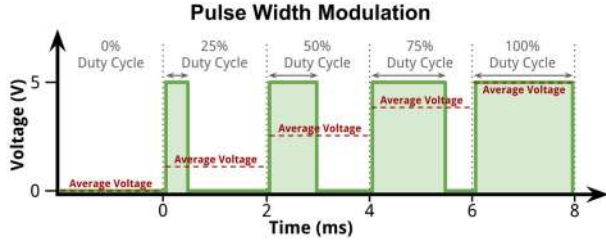
10 PWM

1 Mart 2021 Pazartesi 23:26

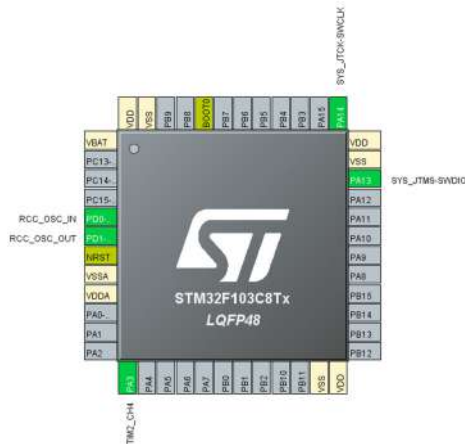
- <https://youtu.be/keS-JecLk7U>
- <https://youtu.be/bXo1TbHVfR4>

10 PWM

Teori



Konfigürasyon Kısmı



Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA3	TIM2_CH4	n/a	Alternate Fu...	n/a	Low		<input type="checkbox"/>

- Pwm için TIM2'den Channel 4 kısmından PWM Generation CH4 seçilir.

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Disable
Channel1	Disable
Channel2	Disable
Channel3	Disable
Channel4	PWM Generation CH4
Combined Channels	Disable
<input type="checkbox"/> Use ETR as Clearing Source	
<input type="checkbox"/> XOR activation	
<input type="checkbox"/> One Pulse Mode	

- 50Hz'lik sinyal üretebilmek için Parameter Settings kısmından Prescaler işlem sonucunu gireriz. Period kısmını Duty Cycle en fazla 100 olduğundan Period kısmına 100-1 olarak gireriz. Pulse kısmı için işlemde 1 kullanırız fakat 1-1'den 0 olarak gireriz. İşlem sonucunda Prescaler 14400 girilir.
- Pulse değeri 50 girilirse doluluk oranı %50 olur.

$$UpdateEvent = \frac{72.000.000}{(Prescaler + 1)(100)} = 50 Hz$$
$$Prescaler + 1 = 14400$$

Counter Settings

Prescaler (PSC - 16 bits value)	14400-1
Counter Mode	Up
Counter Period (AutoReload Register - 1.. 100-1	
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

- Bunun için bir değişiklik yapmıyoruz.

PWM Generation Channel 4

Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

- NVIC Settings'den TIM2 global interrupt'ın Enabled kısmı açık yapılır.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0

- Pin olarak PA3 seçilmiş. Bunu üzerine Ctrl tuşu ile basarsak bize farklı renkte PB11 üzerindede kullanabileceğimizi gösteriyor.

Kod Kısmı

- PWM işlemini başlatıyoruz. Başlatmak için hal.tim.c dosyasından fonksiyonu çekiyoruz.

```
1428 HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)
```

```
98 /* Initialize all configured peripherals */
99 MX_GPIO_Init();
100 MX_TIM2_Init();
101 /* USER CODE BEGIN 2 */
102 HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_4);
103 /* USER CODE END 2 */
```

- Kodumuza fonksiyon ekliyoruz. Burda CubeMX'de PWM için yer alan ayarları yazdık.

```
57= /* Private user code -----
58 /* USER CODE BEGIN 0 */
59=void SetPwm(uint16_t pulseValue)
60 {
61     TIM_OC_InitTypeDef sConfigOC;
62
63     sConfigOC.OCMode=TIM_OCMode_PWM1;
64     sConfigOC.Pulse=pulseValue;
65     sConfigOC.OCpolarity=TIM_OCPolarity_HIGH;
66     sConfigOC.OCFastMode=TIM_OCFAST_DISABLE;
67     HAL_TIM_PWM_ConfigChannel(&htim2,&sConfigOC,TIM_CHANNEL_4);
68     HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_4);
69 }
70 /* USER CODE END 0 */
```

```
45 /* USER CODE BEGIN PV */
46 uint32_t pwm_deger=0;
47 /* USER CODE END PV */
```

- SetPwm komutu ile değişkene değer göndererek atama yapıyoruz.

```
106 /* USER CODE BEGIN WHILE */
107 while (1)
108 {
109     /* USER CODE END WHILE */
110
111     /* USER CODE BEGIN 3 */
112     SetPwm(pwm_deger);
113     pwm_deger++;
114     if( pwm_deger>100)
115     {
116         pwm_deger=0;
117     }
118     HAL_Delay(100);
119 }
120 /* USER CODE END 3 */
```

11 USART

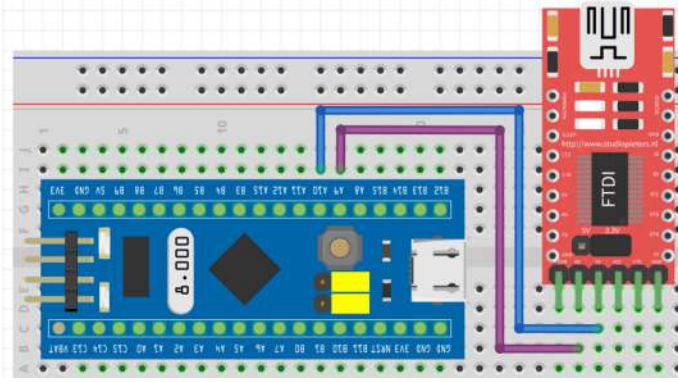
21 Eylül 2021 Salı 19:33

- <https://youtu.be/IFTHz1-cB3I>
- <https://youtu.be/1vipR-6nJaQ>
- <https://youtu.be/a2APUrVwd4>

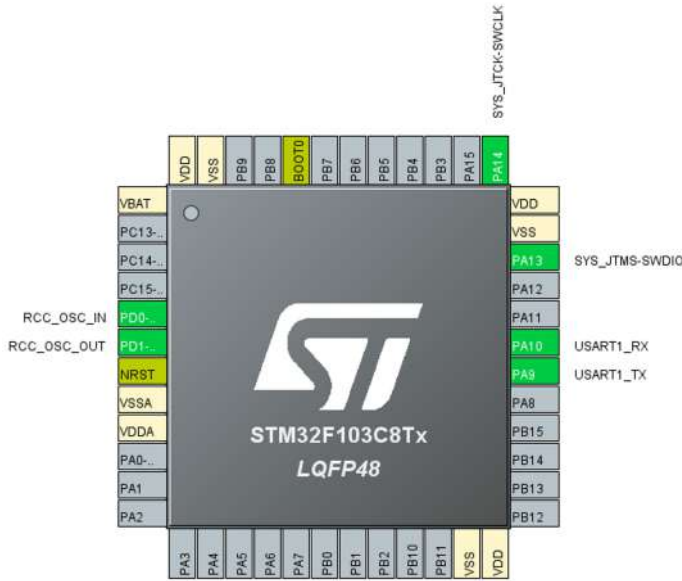
11 USART

Devre Kısmı

- Aynı bilgisayardan güç alındığından GND ortak bağlanmadı.



Konfigürasyon Kısmı



Pin Na...	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA9	USART1_TX	n/a	Alternate F...	n/a	High		<input type="checkbox"/>
PA10	USART1_RX	n/a	Input mode	No pull-up a...	n/a		<input type="checkbox"/>

- Connectivity kısmından USART1 seçilir. Mode kısmında Asynchronous yani Asenkron seçilir. Configuration kısmın Parametre Settings'de bir değişiklik yapmıyoruz.

Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

Data Direction	Receive and Transmit
Over Sampling	16 Samples

- NVIC Settings'den USART1 global interrupt'ın Enabled kısmı açık yapılır.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

Kod Kısmı

- Bu fonksiyona hal_uart.c dosyasından ulaşırız.

```

1124 /**
1125  * @brief Sends an amount of data in blocking mode.
1126  * @note When UART parity is not enabled (PCE = 0), and Word Length is configured to 9 bits (M1-M0 = 01),
1127  * the sent data is handled as a set of u16. In this case, Size must indicate the number
1128  * of u16 provided through pData.
1129  * @param huart Pointer to a UART_HandleTypeDef structure that contains
1130  * the configuration information for the specified UART module.
1131  * @param pData Pointer to data buffer (u8 or u16 data elements).
1132  * @param Size Amount of data elements (u8 or u16) to be sent
1133  * @param Timeout Timeout duration
1134  * @retval HAL status
1135  */
1136 HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)

```

- uint8_t tipinde bir data göndermemizi istiyor fakat biz char değişkeni kullandığımızdan bir çevrim yapmamız gerekiyor.

```

103 HAL_UART_Transmit(&huart1, (uint8_t *) giden_data, 20, 1000);

```

- Gönderdiğimiz karakterin uzunluğunu kendimiz otomatik seçtirebiliriz. 20 yerine strlen deriz ve datanın ismini yazarız. Strlen komutunu tanıması için string kütüphanesinde ekleriz.

```

103 HAL_UART_Transmit(&huart1, (uint8_t *) giden_data, strlen(giden_data), 1000);

```

- Receive yapmak için interrupt kullanacağız.

```

1349 /**
1350  * @brief Receives an amount of data in non blocking mode.
1351  * @note When UART parity is not enabled (PCE = 0), and Word Length is configured to 9 bits (M1-M0 = 01),
1352  * the received data is handled as a set of u16. In this case, Size must indicate the number
1353  * of u16 available through pData.
1354  * @param huart Pointer to a UART_HandleTypeDef structure that contains
1355  * the configuration information for the specified UART module.
1356  * @param pData Pointer to data buffer (u8 or u16 data elements).
1357  * @param Size Amount of data elements (u8 or u16) to be received.
1358  * @retval HAL status
1359  */
1360 HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)

```

- Terminalden yazı yazıp yolladığımızda Debug kısmında gelen_data dizisinde görmüş oluruz.

```

103 HAL_UART_Receive_IT(&huart1, (uint8_t *) gelen_data, 10);

```

```

23 /* Private includes -----
24 /* USER CODE BEGIN Includes */
25 #include "string.h"
26 /* USER CODE END Includes */

```

```

45 /* USER CODE BEGIN PV */
46 char gelen_data[20]="deneme\r\n";
47 char giden_data[20];
48 /* USER CODE END PV */

```

```

96 /* Infinite loop */
97 /* USER CODE BEGIN WHILE */
98 while (1)
99 {
100 /* USER CODE END WHILE */
101
102 /* USER CODE BEGIN 3 */
103 HAL_UART_Receive_IT(&huart1, (uint8_t *) gelen_data, 10);
104 HAL_UART_Transmit(&huart1, (uint8_t *) giden_data, strlen(giden_data), 1000);
105 HAL_Delay(500);
106 }
107 /* USER CODE END 3 */
108 }

```

12 I2C EEPROM

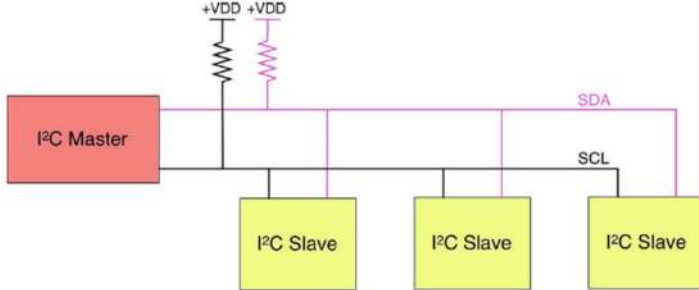
1 Mart 2021 Pazartesi 23:26

- <https://youtu.be/Rzhdm1oV9-I>

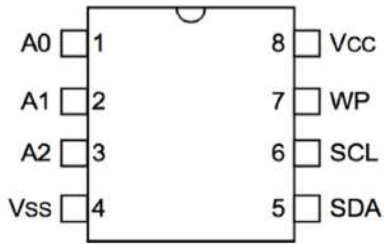
14 I2C EEPROM

Teori Kısmı

- Haberleşmenin tüm hat boyunca hatasız bir şekilde sağlanabilmesi için SDA ve SCL hatları, pull-up dirençlerle VCC hattına bağlanmalıdır. Çoğunlukla 4.7KΩ kullanılır.

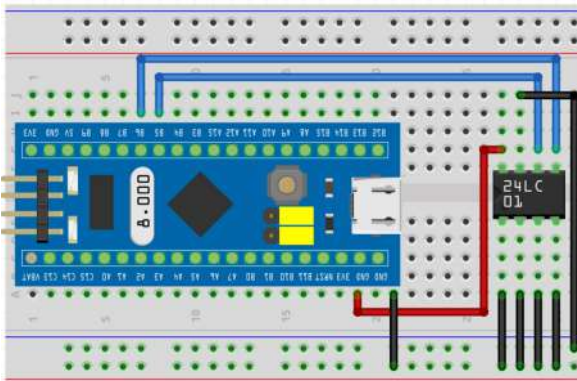


- 24LC64'ün pinout gösterimi aşağıdaki gibidir. M24256'da kullanılabilir.
- A0,A1 ve A2 uçları adresleme uçlarıdır. WP okuma korumasıdır.

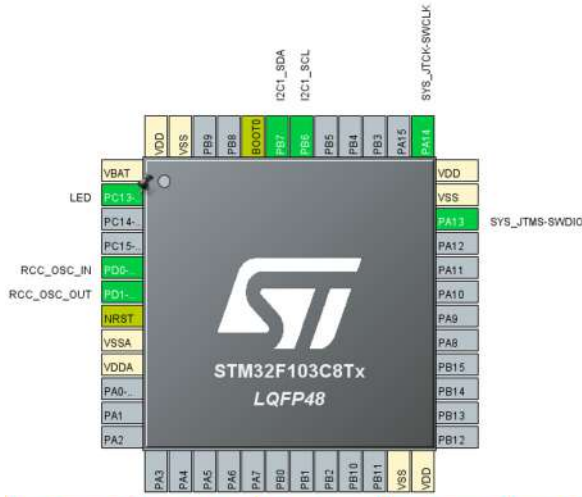


Devre Kısmı

- WP pinini GND'ye bağladık.
- Adresleme uçlarını da GND'ye bağladık.



Konfigürasyon Kısmı



Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull...	Maximum ...	User Label	Modified
PC13-TA...	n/a	Low	Output Pus...	No pull-up ...	Low	LED	<input checked="" type="checkbox"/>

- 72MHz frekans yüksek olduğundan I2C hatların çalışma hızlarını Low yapıyoruz.

Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull...	Maximum ...	User Label	Modified
PB6	I2C1_SCL	n/a	Alternate F...	n/a	Low		<input checked="" type="checkbox"/>
PB7	I2C1_SDA	n/a	Alternate F...	n/a	Low		<input checked="" type="checkbox"/>

- Connectivity kısmından I2C1 seçilir ve sekmeden I2C seçilir. Parametre Settings kısmında bir değişiklik yapmıyoruz.

Master Features

I2C Speed Mode	Standard Mode
I2C Clock Speed (Hz)	100000

Slave Features

Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

Kod Kısmı

```

45 /* USER CODE BEGIN PV */
46 uint8_t giden[5]={50,60,70,80,90};
47 uint8_t gelen[5];
48 /* USER CODE END PV */

```

- Bu fonksiyonlara hal_i2c.c dosyasından ulaşırız. Önce yazma işlemi daha sonra okuma işlemi gerçekleştiriyoruz.

2541 HAL_StatusTypeDef HAL_I2C_Mem_Write(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout)

- DevAddress kısmında A0, A1 ve A2 adresleme uçlarını GND'ye bağladığımızdan adresler a0 olacak. Eğer GND'ye bağlamassak sadece A0'ı VCC'ye bağlarsak 1 eklenmiş gibi olacak böylece adres değişecek.
- MemAdress, hangi adresten itibaren yazmak istiyorsunuz. MemAddSize, ne kadar byte'lık yazacaksınız. pData, yazacağımız datayı yazıyoruz. Size, her seferinde kaç tane yazacağımız. Timeout ise süre yani ne kadar süre sonra bırak anlamındadır.

2664 HAL_StatusTypeDef HAL_I2C_Mem_Read(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData, uint16_t Size, uint32_t Timeout)

```

92 /* USER CODE BEGIN 2 */
93 HAL_Delay(15);
94 HAL_I2C_Mem_Write(&hi2c1,0xa0,0,5,giden,5,10);
95 HAL_Delay(10);
96 HAL_I2C_Mem_Read(&hi2c1,0xa0,0,5,gelen,5,1);
97 /* USER CODE END 2 */

```