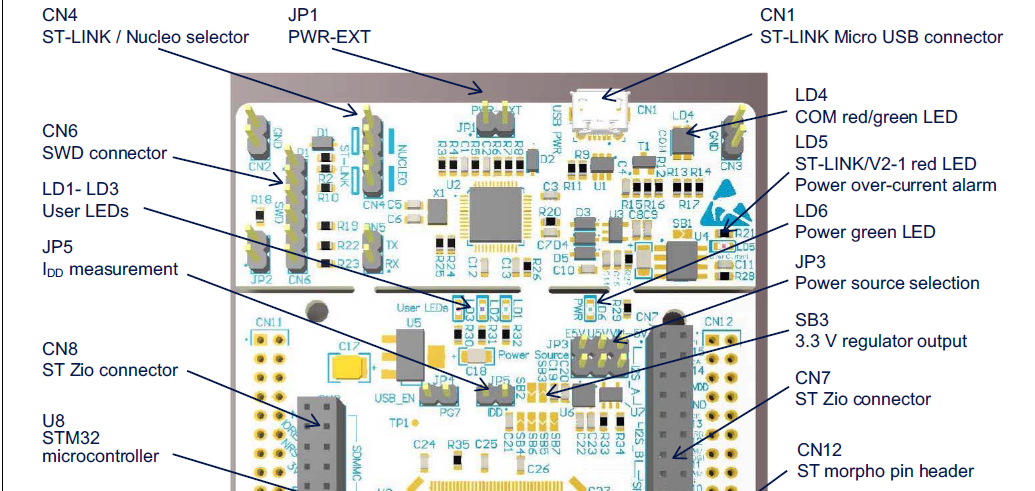
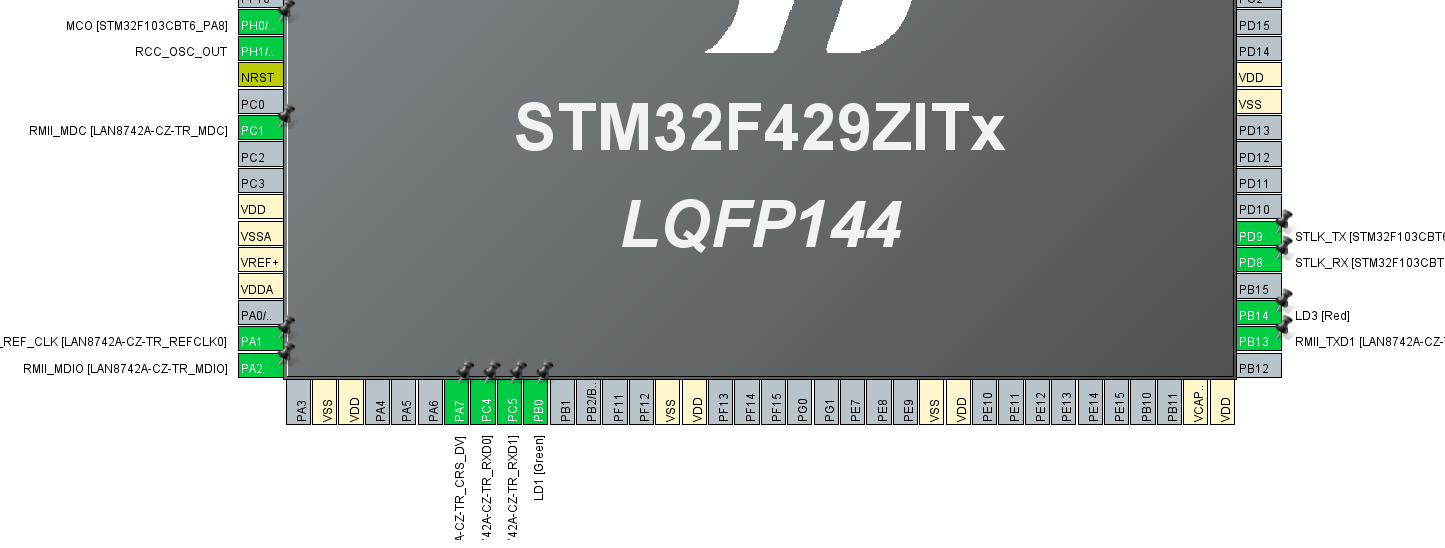
**NUCLEO-F429ZI**

**Örnek Projeler:**

**1) Main loopta LED yakmak**

Yakmak istenilen LED’in pini bulunur.





HAL kütüphanesi kullanılarak pinin durumu koyduğumuz delaye göre toggle edilir.

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_GPIO\_TogglePin(GPIOB, GPIO\_PIN\_0);

HAL\_Delay(500);

}

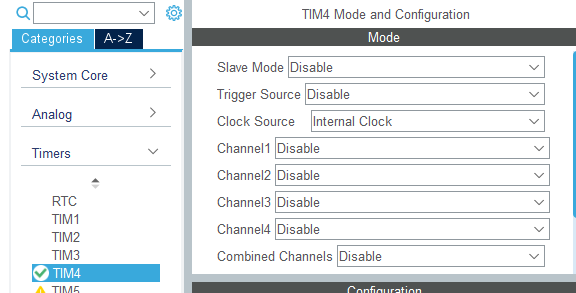
/\* USER CODE END 3 \*/

2) Timer interrupt ile LED yakmak

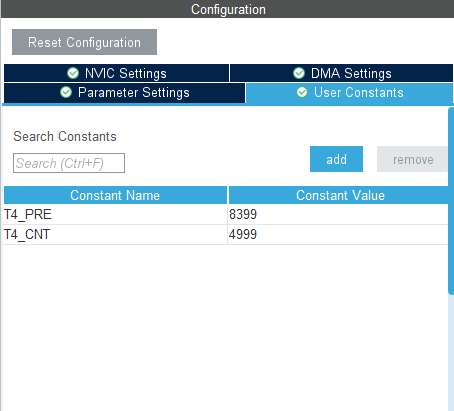
**IOC AYARLARI**

TIM4 seçilir.

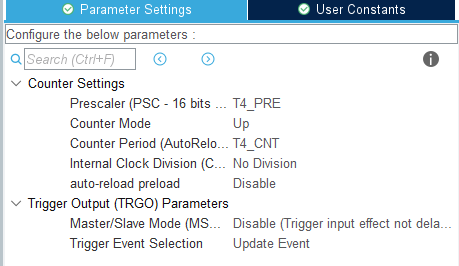
Clock source Internal Clock yapılır.



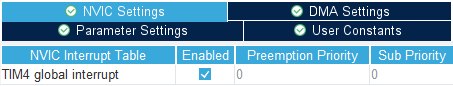
User constants menüsünden T4\_PRE ve T4\_CNT constantları istenilen süre için formüle göre((ADB1/(T4\_PRE+1)) \*(T4\_CNT+1)) ayarlanır.



Parameter menüsünden Prescaler’a T4\_PRE, Counter Period’a T4\_CNT yazılır.



Son olarak NVIC Settingsden TIM4 global interrupt enable hale getirilir.



/\* USER CODE BEGIN 0 \*/

**void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim){

**if**(htim->Instance == TIM4){

HAL\_GPIO\_TogglePin(GPIOB, GPIO\_PIN\_14);

}

}

/\* USER CODE END 0 \*/

Bu fonksiyon timer’ı ayarladığımız süreye göre((ADB1/(T4\_PRE+1)) \*(T4\_CNT+1)) her süre bitiminde çağırılır. Bizim örneğimiz için (ADB1 = 84MHz, T4\_PRE = 8399, T4\_CNT = 4999) her bir saniyede bir.

/\* USER CODE BEGIN 2 \*/

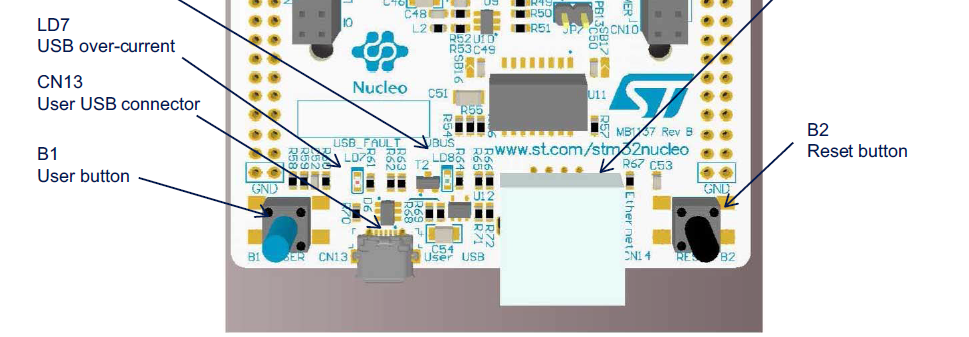
HAL\_TIM\_Base\_Start\_IT(&htim4);

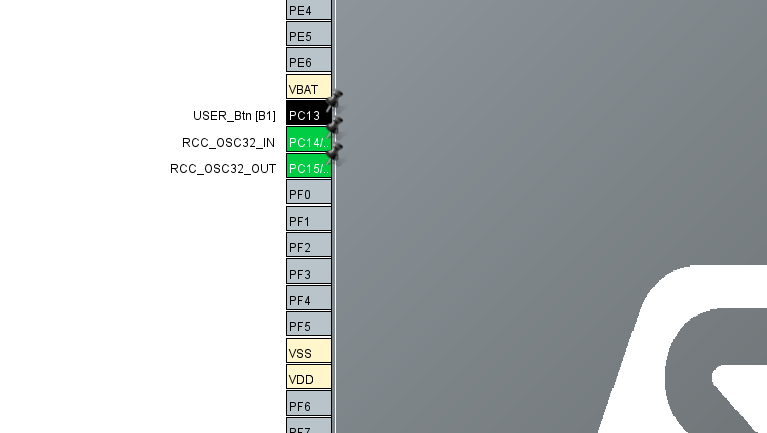
/\* USER CODE END 2 \*/

Bu kod timer’I başlatır.

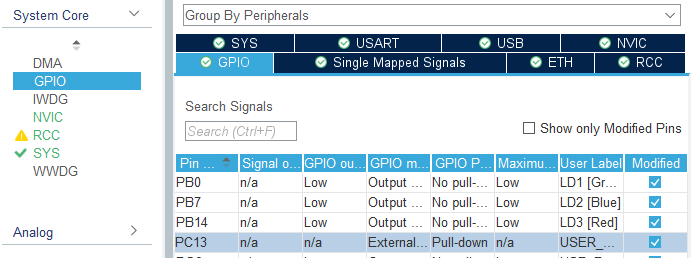
**3) Button kullanımı**

**IOC Ayarları**

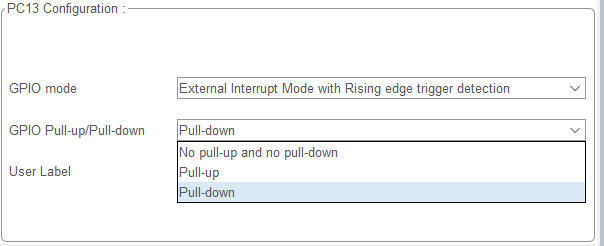




Butonun olduğu port GPIO\_Input moduna alınır.



GPIO ayarlarından pull-down moduna alınır.



/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_Delay(20);

**if**(HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_13)==1){

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_7,*SET*);

}

HAL\_Delay(480);

}

/\* USER CODE END 3 \*/

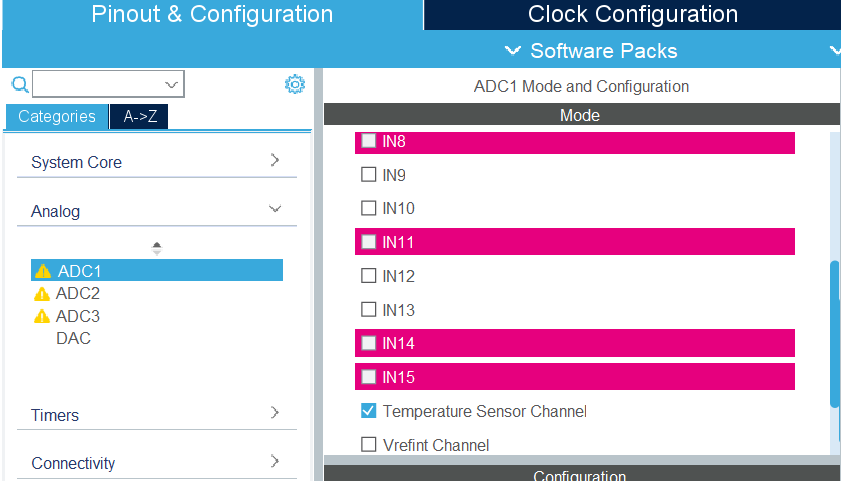
Bu kodla butonun değeri okunarak basıldıysa istediğimiz LED’i yakar. 20ms delay koymamızın sebebi butona basılırken oluşan software debouncingi önlememiz.

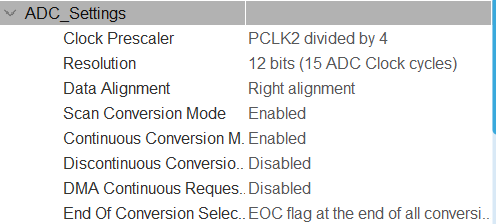
**4)ADC kullanımı ve iç sıcaklığı okuma**

**Interrupt Callback ile:**

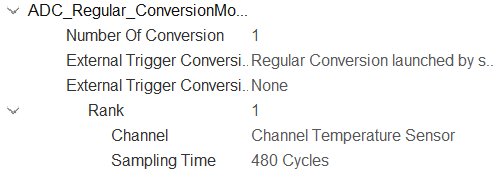
**IOC Ayarları**

PIN ADC\_IN ayarlanır

 İşlemcide bulununan iç sıcaklık sensörü seçilir.

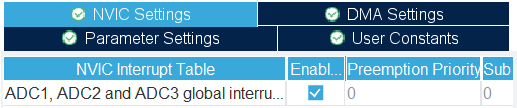


Parameter ayarlarından Scan Conversion Mode ve Continuous Conversion Mode enable hale getirilir. End Of Conversion all moduna getirilir.

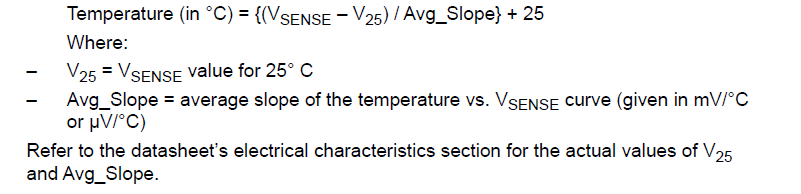


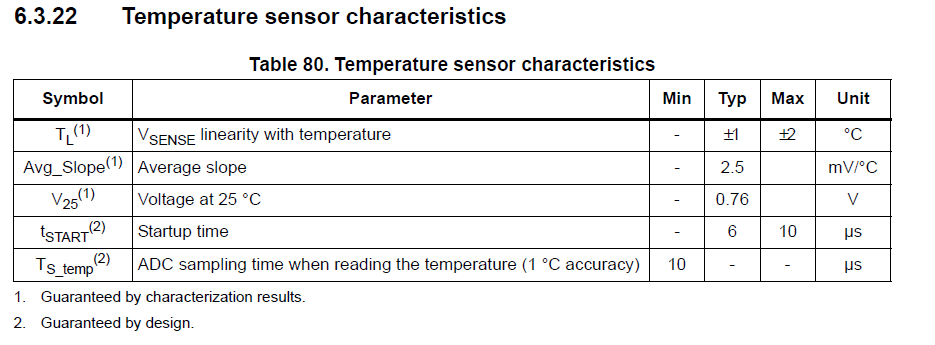
İstenilen kanallar seçilir.

Sampling time ayarlanır.



NVIC ayarlarından global interrupt enable hale getirilir.





/\* USER CODE BEGIN PV \*/

uint32\_t ADC\_deger[1];

**float** temp;

**float** celcius;

uint32\_t ADC\_okunanData;

/\* USER CODE END PV \*/

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_IT(&hadc1);

/\* USER CODE END 2 \*/

Interrupt’ı başlatır.

**void** **HAL\_ADC\_ConvCpltCallback**(ADC\_HandleTypeDef\* hadc)

{

ADC\_deger[0]=HAL\_ADC\_GetValue(&hadc1);

ADC\_okunanData= ADC\_deger[0] & 0xFFFF;

temp = ADC\_okunanData / 4095.0 \* 3.3;

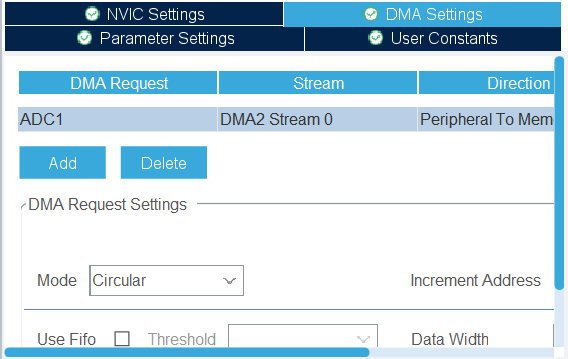
celcius = (temp - 0.76)/0.0025+25;

}

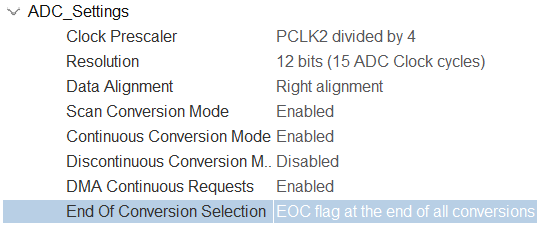
Interrupt callback fonksiyonunun içinde datasheette yazan verilere göre celcius cinsinden sıcaklık hesaplanır.

**DMA ile:**

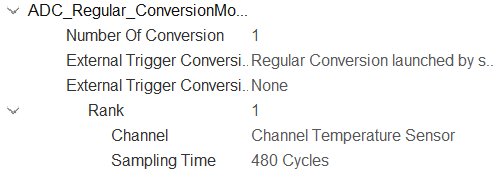
**IOC Ayarları**

 Interrupt callbackte olduğu gibi ADC1 den Temp Sensor aktif edilir.

DMA settingsden dma eklenir ve circular moda alınır.



Parametreler resimdeki gibi ayarlarnır.



İstenilen kanallar seçilir.

Sampling time ayarlanır.

/\* USER CODE BEGIN PV \*/

**float** temp;

**float** celcius;

uint32\_t ADC\_deger[1];

uint32\_t ADC\_okunanData;

/\* USER CODE END PV \*/

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_DMA(&hadc1, ADC\_deger, 1);

/\* USER CODE END 2 \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

ADC\_okunanData = ADC\_deger[0] & 0xFFFF;

temp = ADC\_okunanData / 4095.0 \* 3.3;

celcius = (temp - 0.76)/0.0025+25;

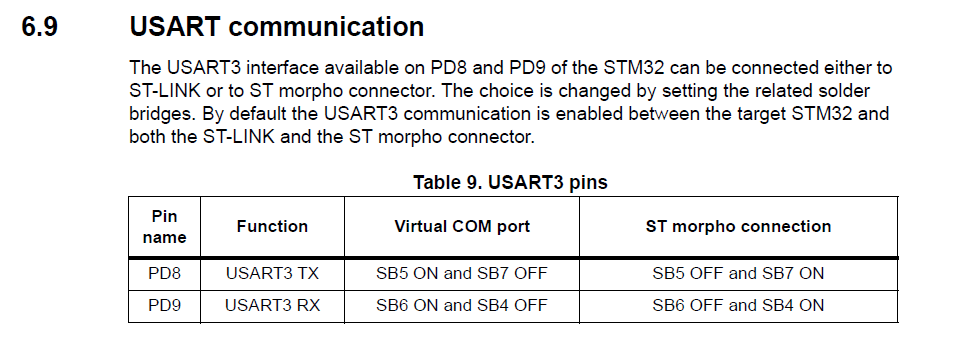
}

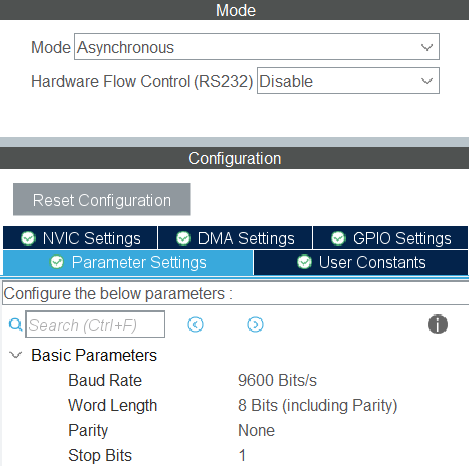
/\* USER CODE END 3 \*/

**5)Seri kanal mesaj alıp-göndermek**

**IOC Ayarları**

İstenilen UART veya USART seçilir. Bizim durumumuzda ST-LINK COM portu ile haberleşeceğimiz için USART3’ü seçiyoruz.



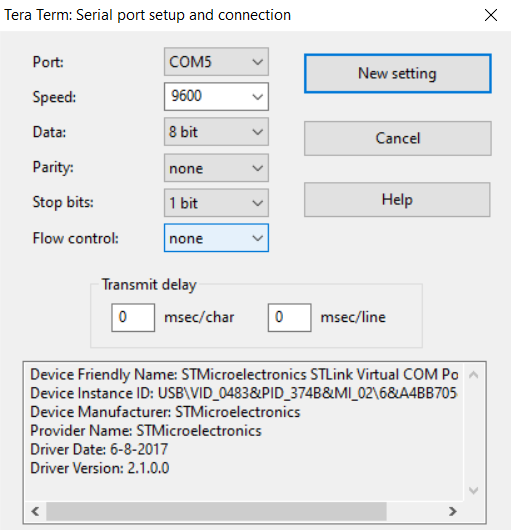


Modunu Asynchronous yapıyoruz.

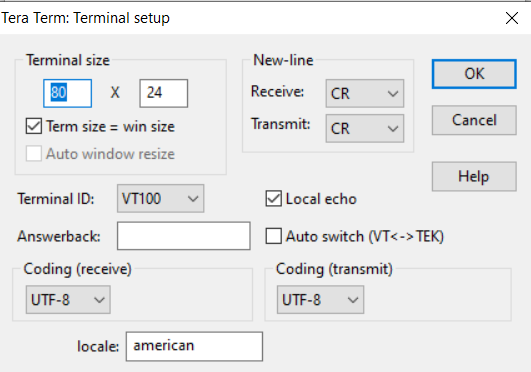
Baud rate’i 9600 veya 115200 yapıyoruz. Bu örnekte 9600.

**Tera Term Ayarları**

Setupdan serial port… seçilir.



Yukarıdaki ayarlar yapılır.



Tekrar setupdan terminale girilir ve locak echo açılır.

/\* USER CODE BEGIN PV \*/

**char** gelen[1];

**char** giden[] = "Test \r\n";

/\* USER CODE END PV \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_UART\_Receive(&huart3, gelen, 1, 10);

**if** (gelen[0]=='s')

{

HAL\_UART\_Transmit(&huart3, giden, **sizeof**(giden), 10);

gelen[0]=0;

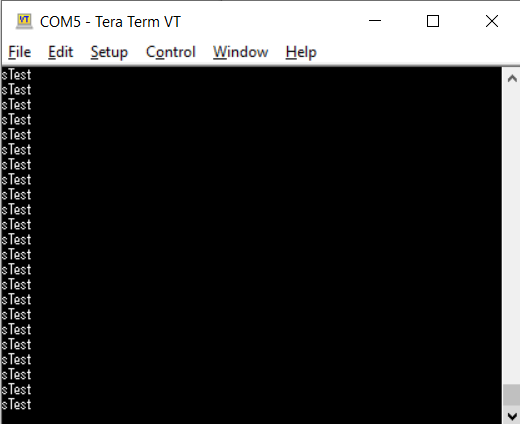
}

}

/\* USER CODE END 3 \*/

Bu kodda Tera Term’den ‘s’ tuşuna basarsak hazırladığımız “Test \r\n” yazısı terminale basılır.

Sonrasında kod başlatıldıktan sonra her s tuşuna basıldığında Test yazısı gelir.

****

**6)İç sıcaklık ve UART kullanarak uygulama**

**IOC Ayarları**

4. örnekteki ADC1 ve DMA aynen ayarlanır.

5. örnekteki USART3 aynen ayarlanır.

**Tera Term Ayarları**

5. örnekteki gibi ayarlanır.

/\* USER CODE BEGIN PV \*/

**float** temp;

**float** celcius;

uint32\_t ADC\_deger[1];

uint32\_t ADC\_okunanData;

**char** gelen[1];

**char** tempDataAscii[8];

/\* USER CODE END PV \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**char** **intToAscii**(**int** digit){

**char** asciiChar;

asciiChar = digit + '0';

**return** asciiChar;

}

/\* USER CODE END 0 \*/

Bu fonksiyon aldığımız celcius verisini istediğimiz gibi bastırmamız için gelen değerleri ascii değerlerine eşitliyor.

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_DMA(&hadc1, ADC\_deger, 1);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

ADC\_okunanData = ADC\_deger[0] & 0xFFFF;

temp = ADC\_okunanData / 4095.0 \* 3.3;

celcius = (temp - 0.76)/0.0025+25;

HAL\_UART\_Receive(&huart3, gelen, **sizeof**(gelen), 10);

**if** (gelen[0] == 's'){

**char** a;

**if**(celcius>0)

{

a = '+';

}

**else** **if**(celcius<0)

{

a = '-';

}

**int** b = celcius\*10;

**int** c = b % 1000;

b = b / 1000;

b = intToAscii(b);

**int** d = c % 100;

c = c / 100;

c = intToAscii(c);

**int** f = d % 10;

f = intToAscii(f);

d = d / 10;

d = intToAscii(d);

**char** e = '.';

//a b c d e f

tempDataAscii[0] = ' ';

tempDataAscii[1] = a;

tempDataAscii[2] = b;

tempDataAscii[3] = c;

tempDataAscii[4] = d;

tempDataAscii[5] = e;

tempDataAscii[6] = f;

HAL\_UART\_Transmit(&huart3,tempDataAscii, **sizeof**(tempDataAscii),10);

HAL\_UART\_Transmit(&huart3, " \370C \r\n", 6, 10);

gelen[0]=0;

}

}

/\* USER CODE END 3 \*/

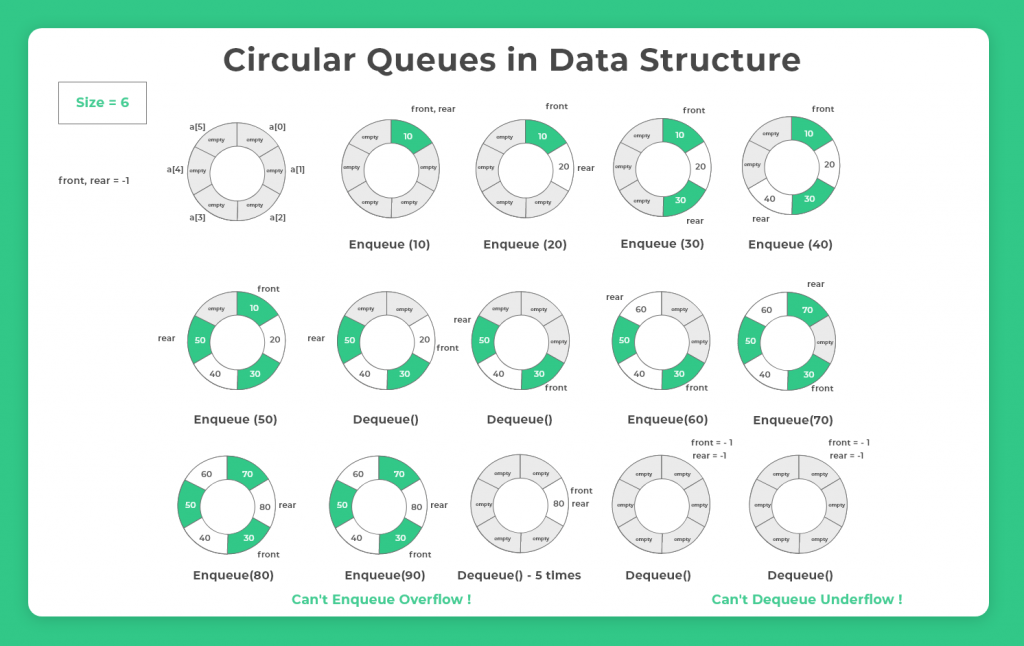
}

Bu kodda aldığımız ADC verisini istediğimiz formatta Tera Term’e basıyoruz.

**7)Uart interrupt queue ve DMA queue ile lower case to upper case projesi**

**Circular queue nedir?**

Sürekli başa dönen bir liste.

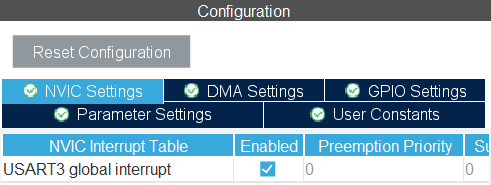
First in first out(FIFO) mantığıyla çalışır. Bu yöntemi kullanmamızın nedeni elimizde büyük bir veri varsa bunu interrupt içinde işlem yaparken oluşan gecikmeden dolayı veri kaybetme şansımız yüksektir. Bu yüzden interrupt içinde sadece veriyi bir listeye kaydederek while döngüsünde sırayla işleriz. Bu sayede veri kaybetmeyiz.

**Interrupt ile circular queue:**

**IOC Ayarları**

USART3 Asynchronous moda alınır.

NVIC ayarlarından USART3 global interrupt enable hale getirilir.

****

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

**#define** SIZE 256

/\* USER CODE END PD \*/

/\* USER CODE BEGIN PV \*/

**char** gelen[1];

**char** giden[1];

**char** items[SIZE];

**int** queueFirst = -1, queueLast = -1;

/\* USER CODE END PV \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**char** **upperToLowerCase**(**int** digit)

{

**char** asciiChar;

asciiChar = digit + ' ';

**return** asciiChar;

}

// Check if the queue is full

**int** **isFull**()

{

**if** ((queueFirst == queueLast + 1) || (queueFirst == 0 && queueLast == SIZE - 1)) **return** 1;

**return** 0;

}

// Check if the queue is empty

**int** **isEmpty**()

{

**if** (queueFirst == -1) **return** 1;

**return** 0;

}

// Check if the

**int** **queueEmpty**()

{

**if**(items[queueFirst] == '\000') **return** 1;

**return** 0;

}

// Adding an element

**void** **enQueue**(**char** element)

{

**if** (isFull())

{

//Queue is full

}

**else**

{

**if** (queueFirst == -1) queueFirst = 0;

queueLast = (queueLast + 1) % SIZE;

items[queueLast] = element;

//Inserted

}

}

// Removing an element

**char** **deQueue**()

{

**char** element;

**if** (isEmpty())

{

//Queue is empty

**return** (-1);

}

**else**

{

element = items[queueFirst];

**if** (queueFirst == queueLast)

{

queueFirst = -1;

queueLast = -1;

}

// Q has only one element, so we reset the queue after dequeing it. ?

**else**

{

items[queueFirst]='\000';

queueFirst = (queueFirst + 1) % SIZE;

}

//Deleted element

**return** element;

}

}

/\* USER CODE END 0 \*/

/\* USER CODE BEGIN 2 \*/

HAL\_UART\_Receive\_IT(&huart3, gelen, **sizeof**(gelen));

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**if**(!queueEmpty())

{

giden[0] = upperToLowerCase(deQueue());

HAL\_UART\_Transmit(&huart3, giden, 1, 10);

}

HAL\_Delay(10);

}

/\* USER CODE END 3 \*/

}

**void** **HAL\_UART\_RxCpltCallback**(UART\_HandleTypeDef \*huart)

{

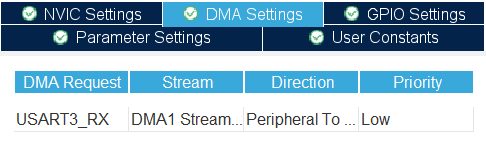
enQueue(gelen[0]);

HAL\_UART\_Receive\_IT(&huart3, gelen, **sizeof**(gelen));

}

**DMA ile circular queue:**

**IOC Ayarları**

****

USART3’e DMA eklenir.

NVIC’den interruptta açtığımız global interrupt kapatılır.

**Kodda yapılan değişiklikler**

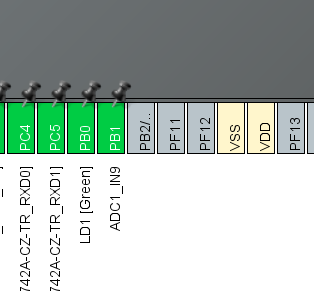
**HAL\_UART\_Receive\_IT**(&huart3, gelen, **sizeof**(gelen));

**HAL\_UART\_Receive\_DMA**(&huart3, gelen, **sizeof**(gelen));

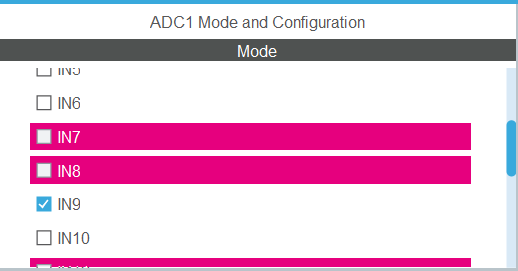
IT’ler DMA ile değiştirilir.

**8)IR Sensör ile mesafe algılama**

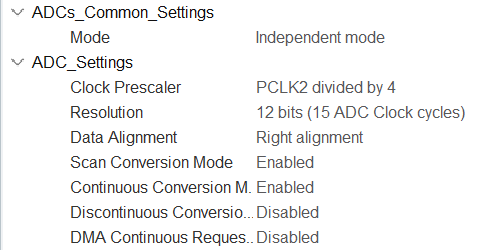
**IOC Ayarları**

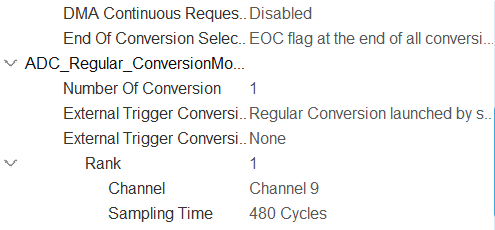


PB1’i ADC\_IN olarak ayarlanır.

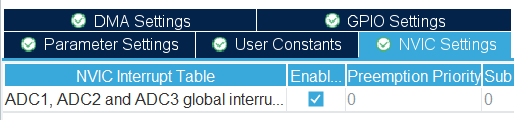


ADC1 ayarlarından in9 seçilir.





Configuration ayarları görseldeki gibi ayarlanır.



NVIC ayarlarından adc1 global interrupt enable hale getirilir.

**Kodlar**

/\* USER CODE BEGIN PV \*/

**float** adc;

**unsigned** **short** ir;

/\* USER CODE END PV \*/

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_IT(&hadc1);

/\* USER CODE END 2 \*/

**void** (**HAL\_ADC\_ConvCpltCallback**(ADC\_HandleTypeDef\* hadc))

{

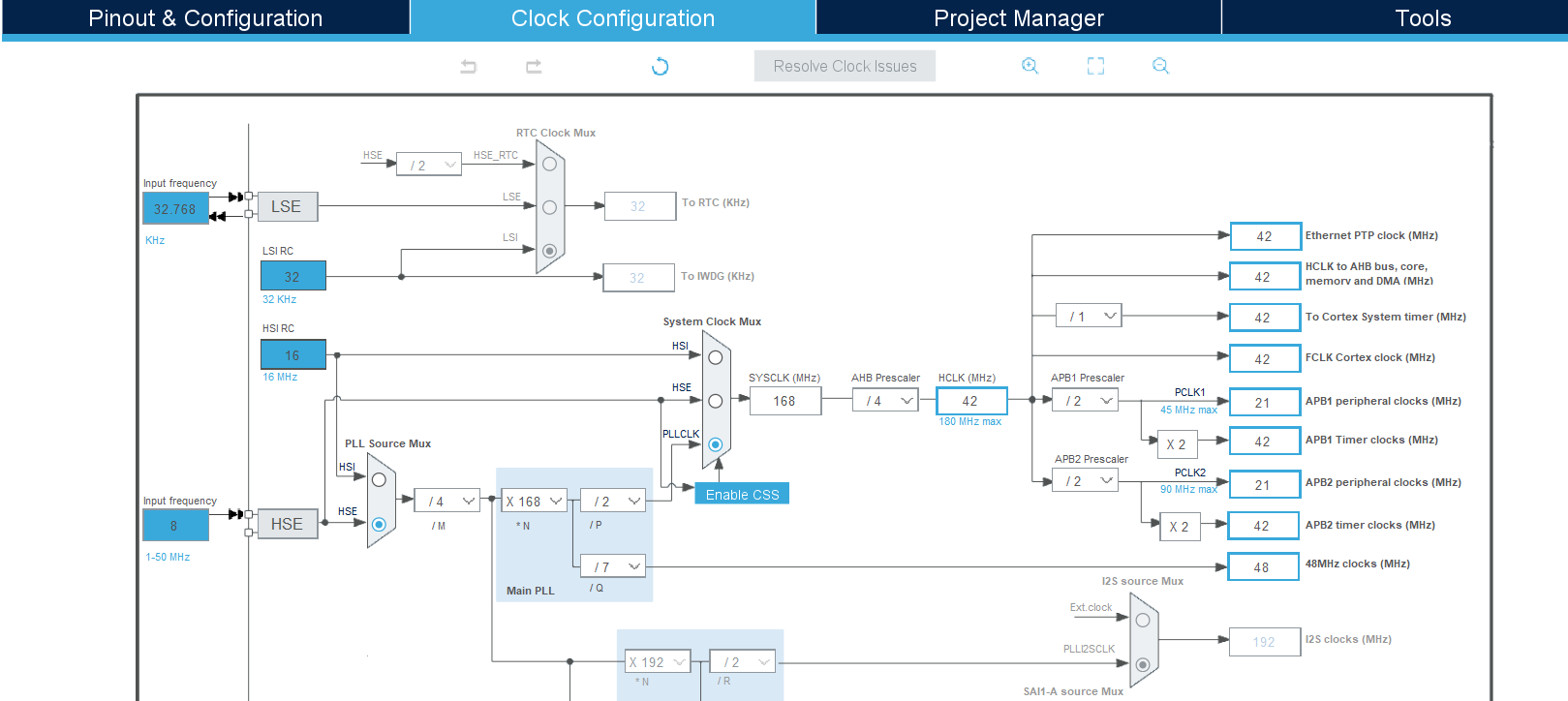
ir = HAL\_ADC\_GetValue(&hadc1);

adc = ir/4095.0 \*3.3;

}

**9)SPI ile tek boardda haberleşme (uart interrupt queue)**

**IOC Ayarları**

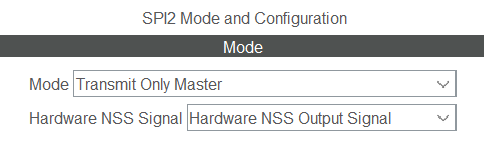
****

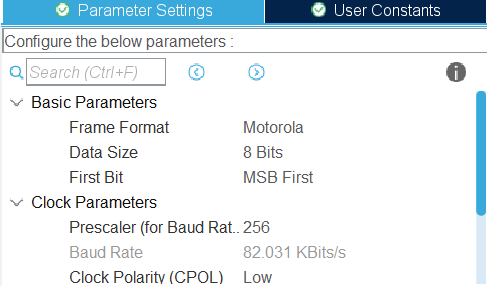
Clock ayarlarından HCLK 42 Mhz’ye ayarlanır. SPI çok hızlı bir haberleşme olduğundan ve jumper kablo kullandığımız için hızını olabildiğince yavaşlatmamız gerek.(ek olarak SPI’ın da scale ayarını buna göre ayarlıyacağız

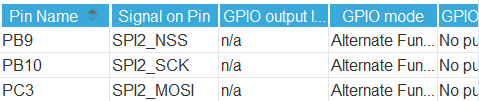
USART3 asynchronous moda alınır.

NVIC den USART3 interrupt enable hale getirilir.

Master modda kullanacağımız SPI2’nin ayarları.

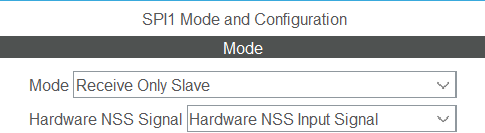
****

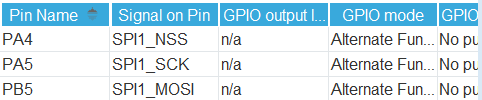
****

****

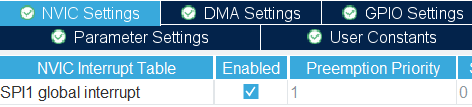
GPIO ayarları.

Slave modda kullanacağımız SPI1’in ayarları.

****

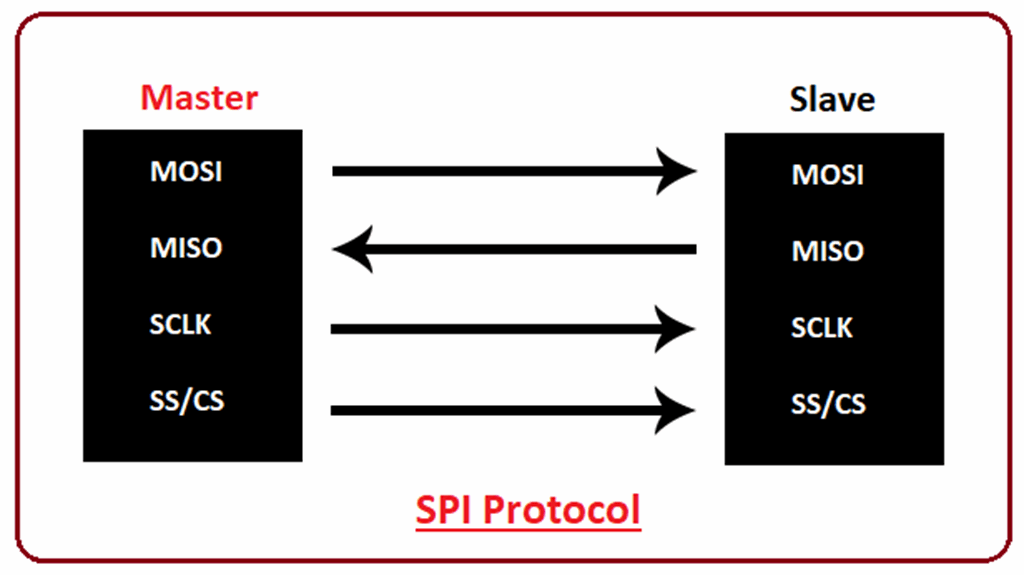
****

GPIO ayarları.



NVIC ayarları.

TeraTerm önceki örneklerdeki gibi ayarlanır.



Son olarak ayarladığımız GPIO pinleri protokole uygun olarak jumper kablolar ile boardda uygun şekilde bağlanır.(transmit/recieve only seçeneklerini kullandığımız için MOSI pinlerine ihtiyaç duymuyoruz)

**Kodlar**

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

**#define** SIZE 256

/\* USER CODE END PD \*/

/\* USER CODE BEGIN PV \*/

**int** it=0;

uint8\_t gelen[1];

uint8\_t giden[1];

uint8\_t gelenspi[1];

uint8\_t gidenspi[1];

**char** items[SIZE];

**int** queueFirst = -1, queueLast = -1;

/\* USER CODE END PV \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

// Check if the queue is full

**int** **isFull**()

{

**if** ((queueFirst == queueLast + 1) || (queueFirst == 0 && queueLast == SIZE - 1)) **return** 1;

**return** 0;

}

// Check if the queue is empty

**int** **isEmpty**()

{

**if** (queueFirst == -1) **return** 1;

**return** 0;

}

// Check if the

**int** **queueEmpty**()

{

**if** (queueFirst == -1) **return** 1;

**return** 0;

}

// Adding an element

**void** **enQueue**(**char** element)

{

**if** (isFull())

{

//Queue is full

}

**else**

{

**if** (queueFirst == -1) queueFirst = 0;

queueLast = (queueLast + 1) % SIZE;

items[queueLast] = element;

//Inserted

}

}

// Removing an element

**char** **deQueue**()

{

**char** element;

**if** (isEmpty())

{

//Queue is empty

**return** (-1);

}

**else**

{

element = items[queueFirst];

**if** (queueFirst == queueLast)

{

queueFirst = -1;

queueLast = -1;

}

// Q has only one element, so we reset the queue after dequeing it. ?

**else**

{

items[queueFirst]='\000';

queueFirst = (queueFirst + 1) % SIZE;

}

//Deleted element

**return** element;

}

}

/\* USER CODE END 0 \*/

/\* USER CODE BEGIN 2 \*/

HAL\_UART\_Receive\_IT(&huart3, gelen, **sizeof**(gelen));

HAL\_SPI\_Receive\_IT(&hspi1, gelenspi, **sizeof**(gelen));

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**if**(!queueEmpty())

{

gidenspi[0] = deQueue();

HAL\_Delay(0.25);

HAL\_SPI\_Transmit(&hspi2, gidenspi, **sizeof**(gidenspi), 10);

}

}

/\* USER CODE END 3 \*/

}

**void** **HAL\_SPI\_RxCpltCallback**(SPI\_HandleTypeDef \* hspi)

{

it++;

giden[0]=gelenspi[0];

HAL\_UART\_Transmit(&huart3, giden, **sizeof**(giden), 10);

HAL\_SPI\_Receive\_IT(&hspi1, gelenspi, **sizeof**(gelenspi));

}

**void** **HAL\_UART\_RxCpltCallback**(UART\_HandleTypeDef \*huart)

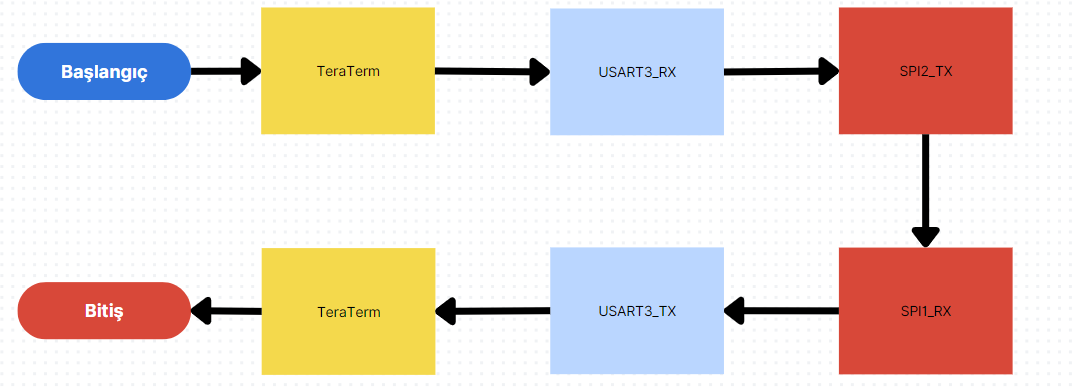
{

//it++;

enQueue(gelen[0]);

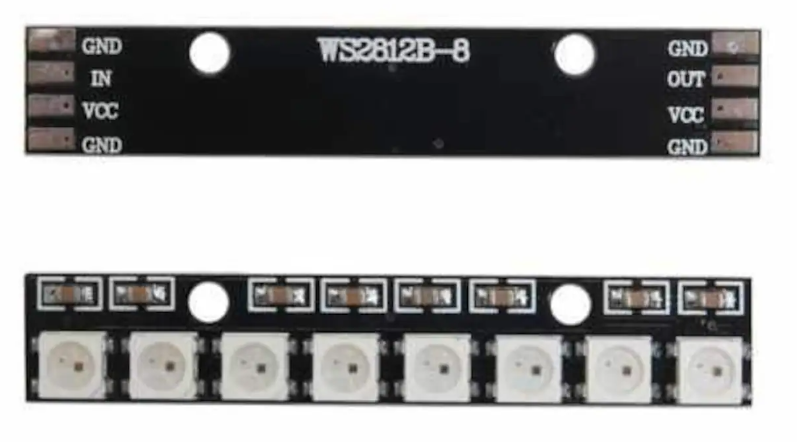
HAL\_UART\_Receive\_IT(&huart3, gelen, **sizeof**(gelen));

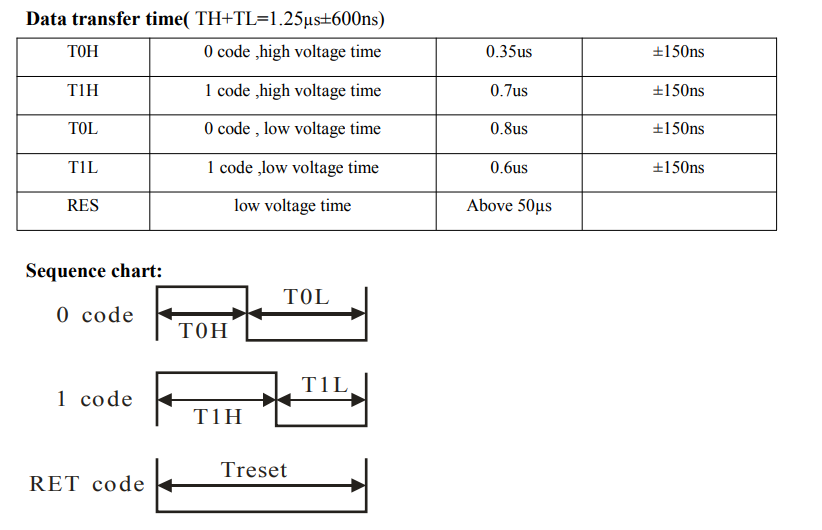
}



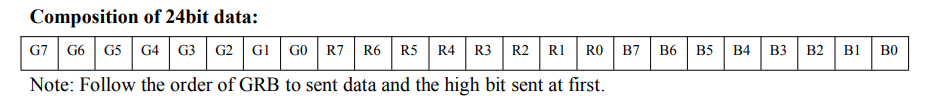
Bu kod sayesinde yukarıdaki adımlar izlenip TeraTerm’e yazdığımız yazıyı kopyası olarak geri alırız.

**10)PWM ve DMA kullanarak WS2812 Ledi kullanmak**

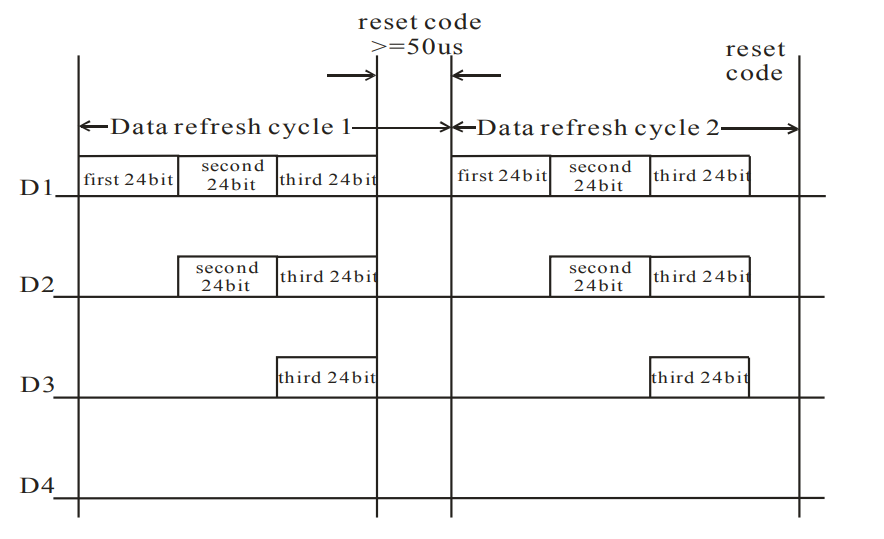
****

****

WS2812’nin kullandığı logic 1, logic 0 ve reset.

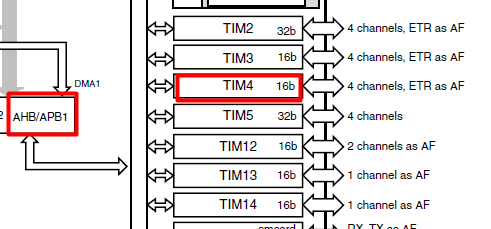
****

Yukarıdaki tanımlanmış logic 1 ve logic 0 sinyallerini üretip GRB sırasında 24bit şekilde hazırladığımız data paketlerini göndermemiz gerek.

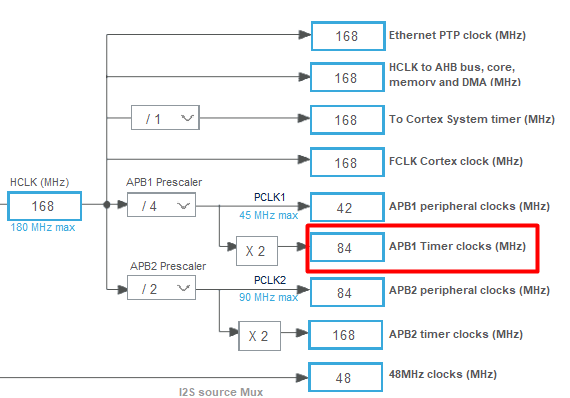


Çalışma mantığı şu şekilde ilk led GRB ye göre düzenlenmiş 24 bitlik renk kodunu alır buna göre yanar. Sonraki 24 biti 2. led alır ve bu şekil devam eder ve reset kodu gelene kadar yanmaya devam eder.

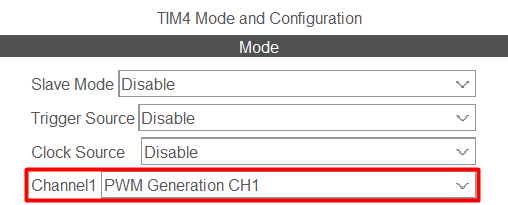
**IOC Ayarları**

****

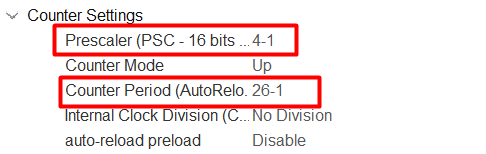
TIM2’den TIM5’e kadar olan general purpose timerlardan istediğimiz seçilir. Bu örnekte TIM4 seçildi.



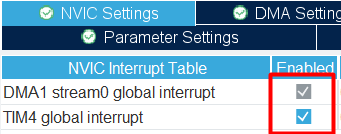
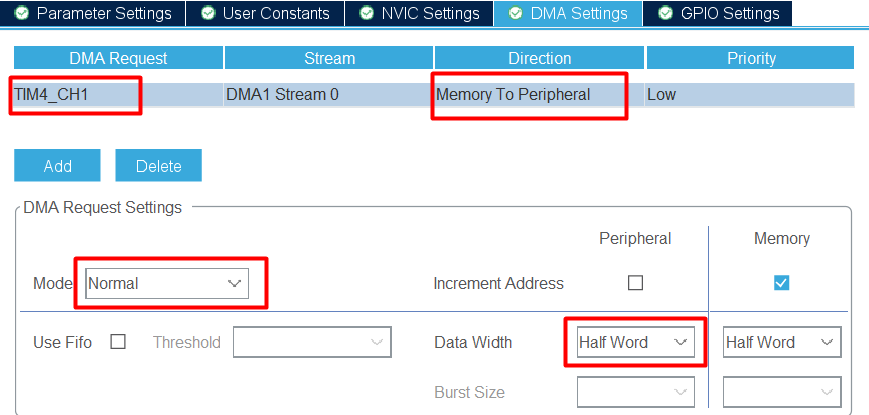
TIM4’ün bağlı olduğu clock bulunur. Bu örnekte APB1(84 MHz).



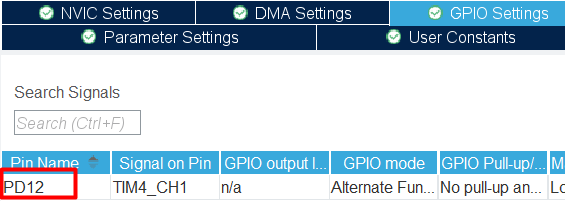
TIM4 Channel1 PWM Generation moduna alınır. Datasheette belirtilen periyod ve duty cycledaki özel dalgaları oluşturmak için Pulse With Modulation(PWM) kullanırız.



PWM dalgamızın periyodunu datasheetteki dalganın periyodu olan 1.25us’ye ayarlamamız gerekiyor. Bu işlem için öncelikle clock değerimiz 84MHz’yi verdiğimiz prescaler değerine(4) böleriz ve 21MHz’yi elde ederiz. 4-1 yazmamızın sebebi ise stm32’nin bu değeri alıp 1 eklemesidir. Aynı şekilde auto reload register’a(ARR) verdiğimiz değer ise prescalerdan sonra kalan değere bölünür 21MHz/26≈807kHz≈1.24us değerine ulaşırız.



DMA kullanmamızın sebebi kullandığımız ekipmanların çok hassas olmasından dolayı(nano saniye mertebesinde) düz PWM kullansaydık registerları manuel olarak ayarlarken kesinlikle bit kaçırıdık. PWM DMA tam da bu durum için kullanılıyor. DMA auarlarında direction ayarını kontrol etmeyi unutmayın hata alırsanız.

****

Son olarak PWM timerın atandığı GPIO pinini bulup led ile kartımızın bağlantılarını yapıyoruz. Led kartını 5V ile besliyoruz. GND ve IN bağlantılarını yapınca bağlantılar tamamlanıyor.

**Kodlar**

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

**typedef** **struct**

{

uint8\_t r;

uint8\_t g;

uint8\_t b;

}rgb\_color;

**typedef** **struct**

{

uint16\_t g[8];

uint16\_t r[8];

uint16\_t b[8];

}rgb\_led;

/\* USER CODE END PTD \*/

Bu structlar işimizi baya kolaylaştırıyor. RGB renk tanımlamak ve led oluşturmak için.

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

**#define** LOGICAL\_ONE 14

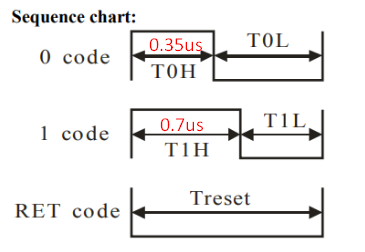
**#define** LOGICAL\_ZERO 7

**#define** NUMBER\_OF\_LEDS 8

/\* USER CODE END PD \*/

Burda logic 1 ve logic 0 değerlerini CCRx değerini değiştirmek için kullanıyoruz. Hesaplamamız için formül (CCRx/ARR)\*Period=duty cycle time

LOGIC\_ONE için CCRx değeri= (0.7us/1.24us)\*25≈14,11 bize tam sayı değeri lazım en yakını olan 14ü alıyoruz.

LOGIC\_ZERO için CCRx değeri= (0.35us/1.24us)\*25≈7,05 bize tam sayı değeri lazım en yakını olan 7yi alıyoruz.

/\* USER CODE BEGIN PV \*/

rgb\_led leds[NUMBER\_OF\_LEDS + 1];

rgb\_color red\_color = {255, 0, 0};

rgb\_color black\_color = {0, 0, 0};

rgb\_color blue\_color = {0, 0, 255};

uint16\_t led\_pos\_counter = 0;

/\* USER CODE END PV \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**void** **reset\_all\_leds**(rgb\_led\* leds,uint16\_t number\_leds)

{

**for**(**int** i=0; i < number\_leds; i++)

{

**for**(**int** j=0; j < 8; j++)

{

(leds + i) ->r[j] = LOGICAL\_ZERO;

(leds + i) ->g[j] = LOGICAL\_ZERO;

(leds + i) ->b[j] = LOGICAL\_ZERO;

}

}

}

**void** **set\_all\_leds**(rgb\_led\* leds,uint16\_t number\_leds)

{

**for**(**int** i=0; i < number\_leds; i++)

{

**for**(**int** j=0; j < 8; j++)

{

(leds + i) ->r[j] = LOGICAL\_ONE;

(leds + i) ->g[j] = LOGICAL\_ONE;

(leds + i) ->b[j] = LOGICAL\_ONE;

}

}

}

**void** **set\_specific\_led**(rgb\_led\* leds, uint16\_t led\_position, rgb\_color color)

{

**for**(**int** j=0; j < 8; j++)

{

**if**(color.r & 0x1 <<j)

{

(leds + led\_position) ->r[7-j] = LOGICAL\_ONE;

}

**else**

{

(leds + led\_position) ->r[7-j] = LOGICAL\_ZERO;

}

**if**(color.g & 0x1 <<j)

{

(leds + led\_position) ->g[7-j] = LOGICAL\_ONE;

}

**else**

{

(leds + led\_position) ->g[7-j] = LOGICAL\_ZERO;

}

**if**(color.b & 0x1 <<j)

{

(leds + led\_position) ->b[7-j] = LOGICAL\_ONE;

}

**else**

{

(leds + led\_position) ->b[7-j] = LOGICAL\_ZERO;

}

}

}

/\* USER CODE END 0 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

//3 adet deneme kodu var istediğiniz kodları commente alarak kullanabilirsiniz.

//1. ledlerin hepsini yakıp söndürme.

reset\_all\_leds(leds, NUMBER\_OF\_LEDS);

HAL\_TIM\_PWM\_Start\_DMA(&htim4, TIM\_CHANNEL\_1, (uint32\_t \*)leds, NUMBER\_OF\_LEDS \* 24 + 24);

HAL\_Delay(300);

set\_all\_leds(leds, NUMBER\_OF\_LEDS);

HAL\_TIM\_PWM\_Start\_DMA(&htim4, TIM\_CHANNEL\_1, (uint32\_t \*)leds, NUMBER\_OF\_LEDS \* 24 + 24);

HAL\_Delay(300);

//2. kara şimşek tarzı kayan kırmızı led.

set\_specific\_led(leds, led\_pos\_counter, black\_color);

led\_pos\_counter++;

**if**(led\_pos\_counter == NUMBER\_OF\_LEDS)

{

led\_pos\_counter = 0;

}

set\_specific\_led(leds, led\_pos\_counter, red\_color);

HAL\_TIM\_PWM\_Start\_DMA(&htim4, TIM\_CHANNEL\_1, (uint32\_t \*)leds, NUMBER\_OF\_LEDS \* 24 + 24);

HAL\_Delay(70);

//3. polis ışıkları.

set\_specific\_led(leds, 0, red\_color);

set\_specific\_led(leds, 1, red\_color);

set\_specific\_led(leds, 2, red\_color);

set\_specific\_led(leds, 3, red\_color);

set\_specific\_led(leds, 4, black\_color);

set\_specific\_led(leds, 5, black\_color);

set\_specific\_led(leds, 6, black\_color);

set\_specific\_led(leds, 7, black\_color);

HAL\_TIM\_PWM\_Start\_DMA(&htim4, TIM\_CHANNEL\_1, (uint32\_t \*)leds, NUMBER\_OF\_LEDS \* 24 + 24);

HAL\_Delay(100);

set\_specific\_led(leds, 0, black\_color);

set\_specific\_led(leds, 1, black\_color);

set\_specific\_led(leds, 2, black\_color);

set\_specific\_led(leds, 3, black\_color);

set\_specific\_led(leds, 4, blue\_color);

set\_specific\_led(leds, 5, blue\_color);

set\_specific\_led(leds, 6, blue\_color);

set\_specific\_led(leds, 7, blue\_color);

HAL\_TIM\_PWM\_Start\_DMA(&htim4, TIM\_CHANNEL\_1, (uint32\_t \*)leds, NUMBER\_OF\_LEDS \* 24 + 24);

HAL\_Delay(100);

}

/\* USER CODE END 3 \*/

}

/\* USER CODE BEGIN 4 \*/

**void** **HAL\_TIM\_PWM\_PulseFinishedCallback**(TIM\_HandleTypeDef \*htim)

{

HAL\_TIM\_PWM\_Stop\_DMA(&htim4, TIM\_CHANNEL\_1);

htim4.Instance->CCR1 = 0;

}

/\* USER CODE END 4 \*/