PROJE ODEVİ

ADC-LCD

BATUHAN TOPAL 17060071

CANER ADSOY 17060032

M.MERT ÇAKIR 17060101

Proje Amacı:ADC kullanarak potansiyometreden okunan değerin LCD ekrana yazdırılması

Projede Kullanılan Malzemeler

- Nucleo-F401RE Geliştirme Kartı
- Potansiyometre
- 2x16 LCD Ekran

Projede Kullanılan Uygulamalar

- STM32 CUBEMX
- Keil UVision 5 IDE
- Proteus 8.9
- STM Studio
- Fritzing

STM32F401RE ADC KULLANIMI

STM32 mikrodenetleyicilerinde ADC işlemi için üç farklı yöntem kullanılmaktadır: PollForConversin, Interrupt ve DMA.

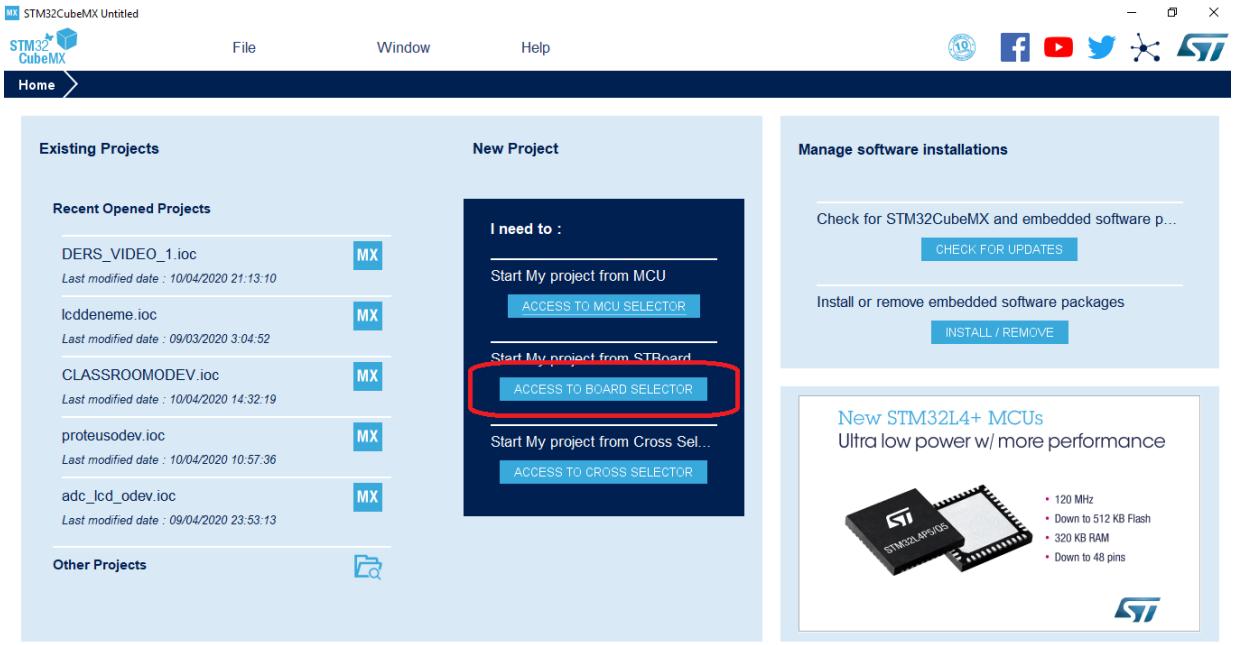
PollForConversion yönteminde ADC ünitesi çevrim işlemini bitirene kadar mikronedenetleyiciyi bloke edilmektedir (blocking).

Interrupt (kesme) yönteminde, ADC ünitesi çevrim işlemlerini tamamlayınca kesme üretmektedir. Bize düşen bu kesmeye servis verip dönüştürülmüş değerleri okumaktır. Kısa da olsa yine mikrodenetleyici meşgul edilmektedir.

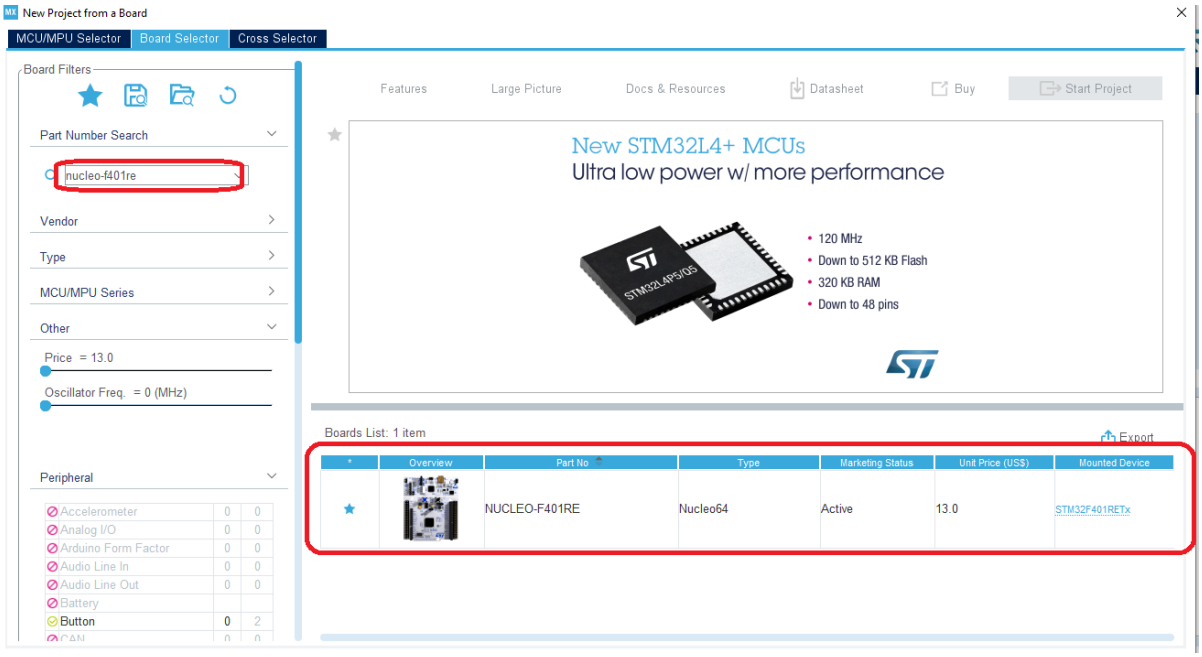
DMA (direct memory access — doğrudan bellek erişimi) yönteminde ADC işleminde dönüştürülen değerler direkt olarak DMA kontrolcüsü tarafından ilgili bellek alanlarına (değişkenlerine) aktarılmaktadır. Geriye kalan sadece ADC değerlerine ihtiyaç duyduğumuz noktada bu değerleri kullanmaktır.

CUBEMX İle ADC OKUMA ve LCD KULLANIMI

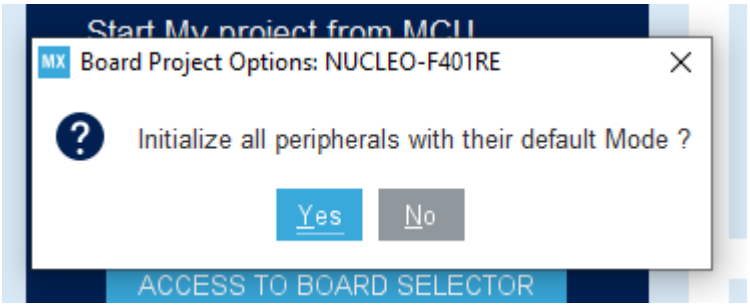
Polling Metodu



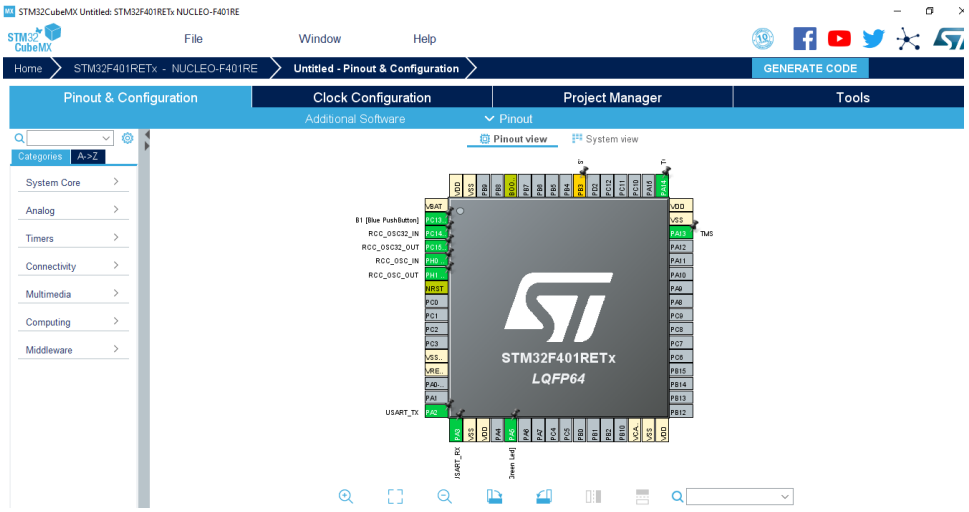
STM32CUBEMX programı başlatılır.Açılan ekranın ortasında bulunan “Access to Board Selector” butonuna basılır.



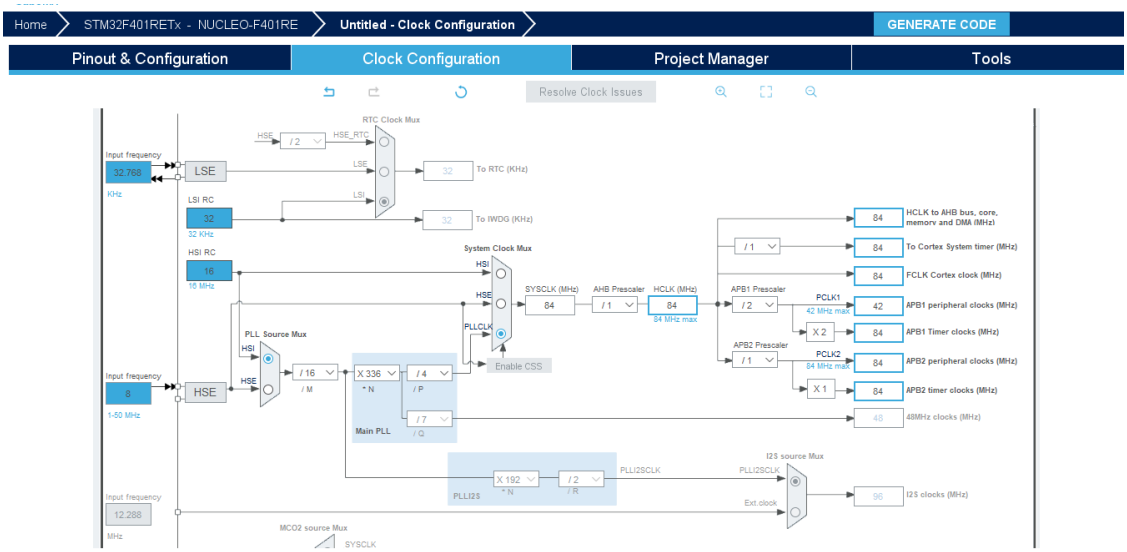
Sol üstteki arama yerine kullanılan geliştirme kartının modeli yazılır.Ekranda kullanılan kartın resmi çıkar.Üzerine 2 defa basılarak açılır.



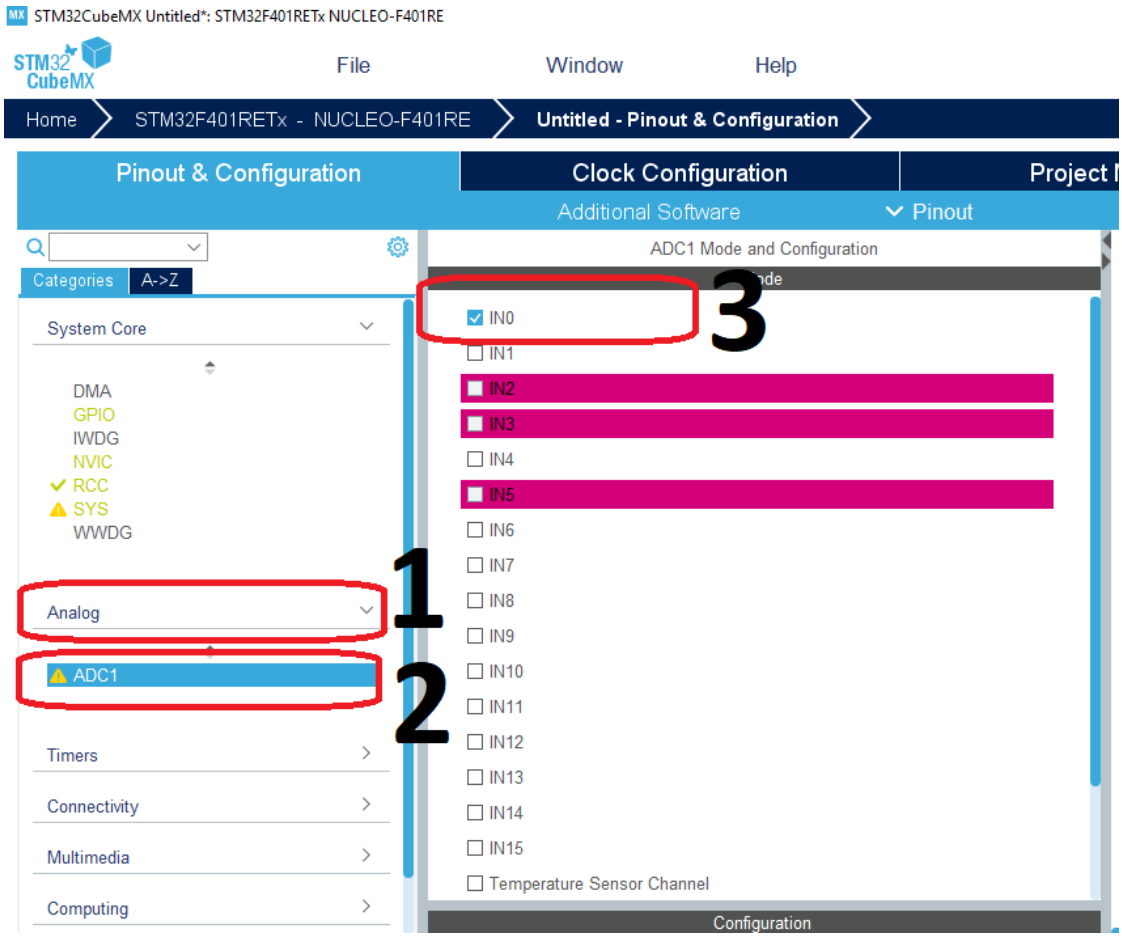
Karşına br uyarı çıkar.Bu uyarıda varsayılan ayarların yapılmasını istiyorsanız **Yes,** istemiyorsanız **No** butonuna basarak açılır.



Açılan ekranı incelendiğinde varsayılan ayarlar olarak kristallerin aktifleştirildiği görülür.Kartın üzerinde bulunan led ve buton tanımlanmıştır.Ayrıca USART haberleşme başlatılmıştır.



Clock Configuration’a gelindiğinde varsayılan ayar olarak STM32F401RE ‘nin maximum hızı olan **84MHz** e ayarlı olduğu görülür.



STM32F401RE serisinde **ADC2** Ve **ADC3** kanalları bulunmaz.

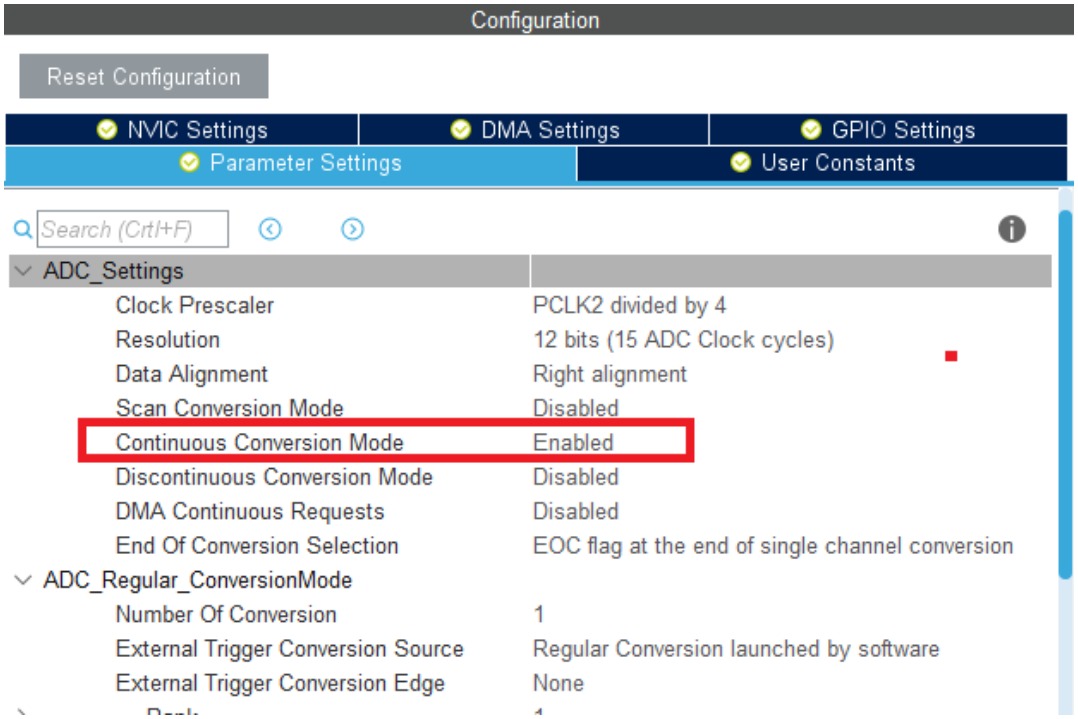
16 harici kanal ile toplamda 19 kanal bulundurur.

12,10,8 ve **6** bitlik çözünürlüğe sahiptir.

2.4-3.6V aralığında **maksimum** hızda;1.8V dan aşağıda yavaş hızda çalışır.

Solda bulunan katagoriler sekmesinden **Analog** altında bulunan ADC1 e basılarak **IN0** seçilir.

Üzeri mor ile çizilmeyen tüm kanallar kullanılabilir durumdadır.Kullanıcı istediğini seçebilir.



Clock Prescaler : Bu ayar ADC' nin çalışmak için ihtiyaç duyduğu saat darbesini PCLK2 hattından kaç ile bölerek kullanacağını belirtir.

Resulation :ADC nin çözünürlük değeridir.12 bitlik okumada 2^12=4096.Yani 0-4095 arasında değerler ölçülür.

Data Alignment: Analog dijital çevrim sonunda elde edilen verinin en değerli biti sağda mı yoksa solda mı olacağını belirler.

Scan Conversion: Aktif edilmesi durumunda aynı ADC' de birden fazla kanal kullanılacağı zaman ölçümleri arka arkaya yapar. Böylece her kanal için ölçüm sonrası ayrı ayrı sayısal dönüştürme yapmaz, aktif edilen tüm kanallar için ölçüm bitince tek sefer dönüştürme yapar. Bu da dönüştürme işleminin daha hızlı olmasını sağlar.

Continuous Conversion: ADC dönüşümü yazılımda bir kere başlatıldığında tekrar başlatmaya gerek kalmaz. Dönüştürme tamamlanınca tekrar çevrim başlar.

DMA Continuous Request: ADC' nin DMA ile kullanımı için yapılması gereken ayar.

Number of Covnersion:ADC başlatıldığında yapılacak çevrim sayısı. Aktif edilmek istenen kanal sayısı kadar seçilmelidir.

Channel: Aktif edilen ADC kanalının seçimi

Sampling Time: ADC ile yapılacak 1 çevrim işlemi için gereken cycle sayısı. Cycle sayısı artarsa çevrim süresi uzar ama ölçüm daha doğru gerçekleşir.

Figure 1. Single-channel, single conversion mode

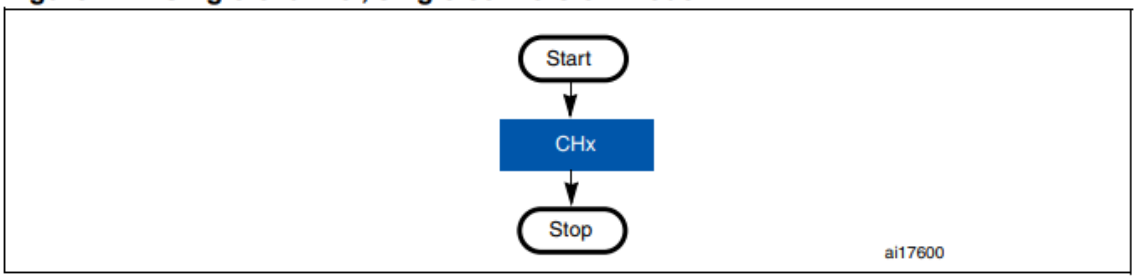
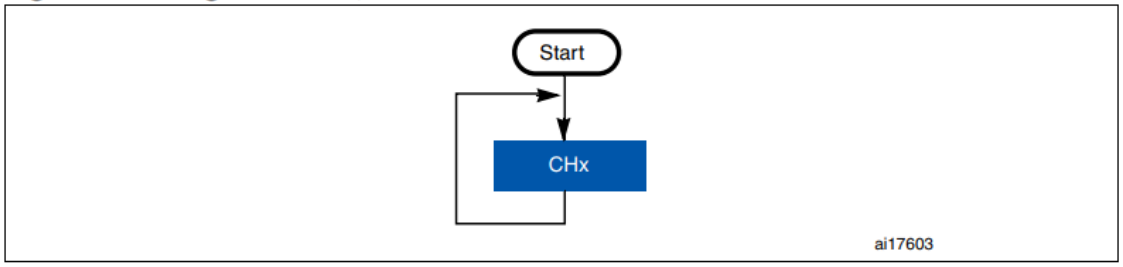
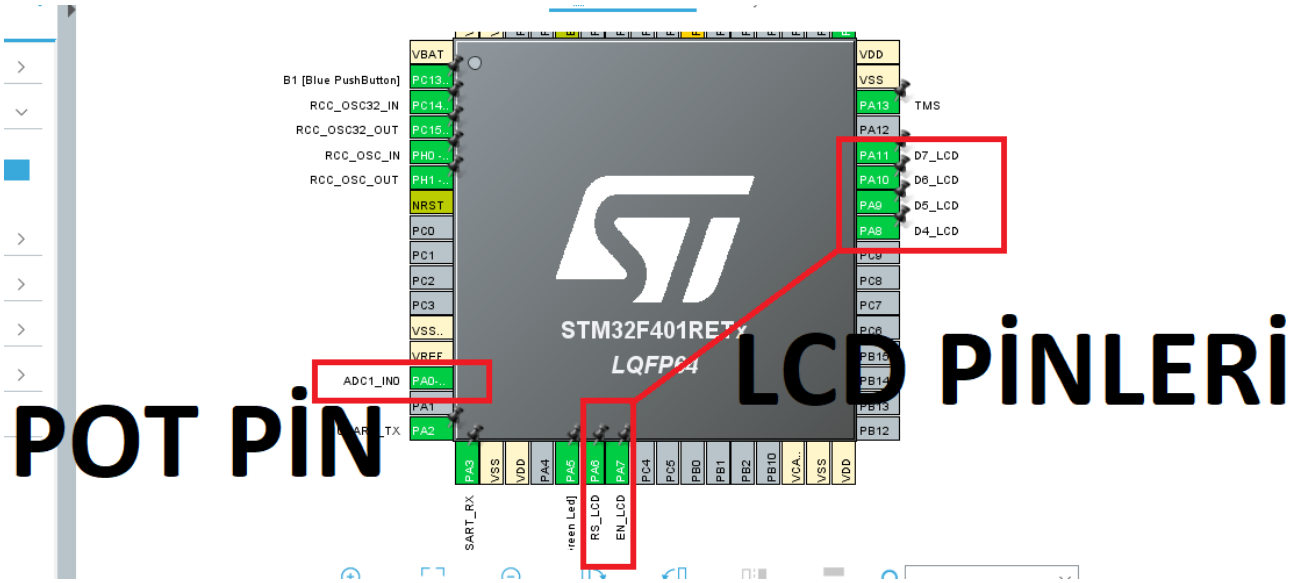


Figure 4. Single-channel, continuous conversion mode

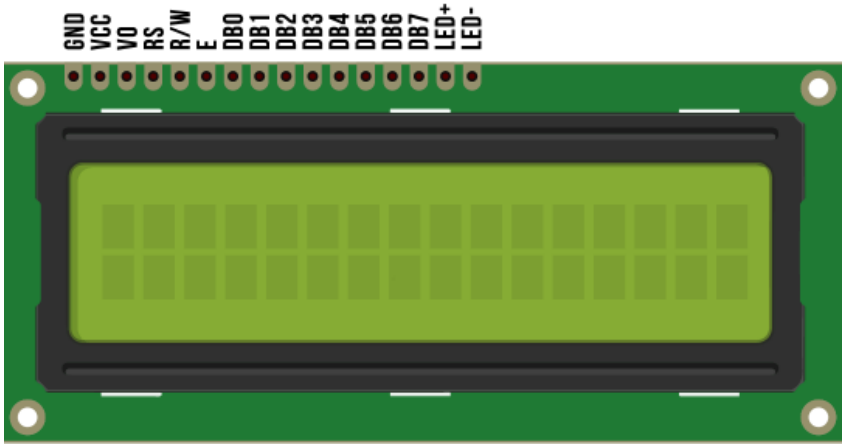


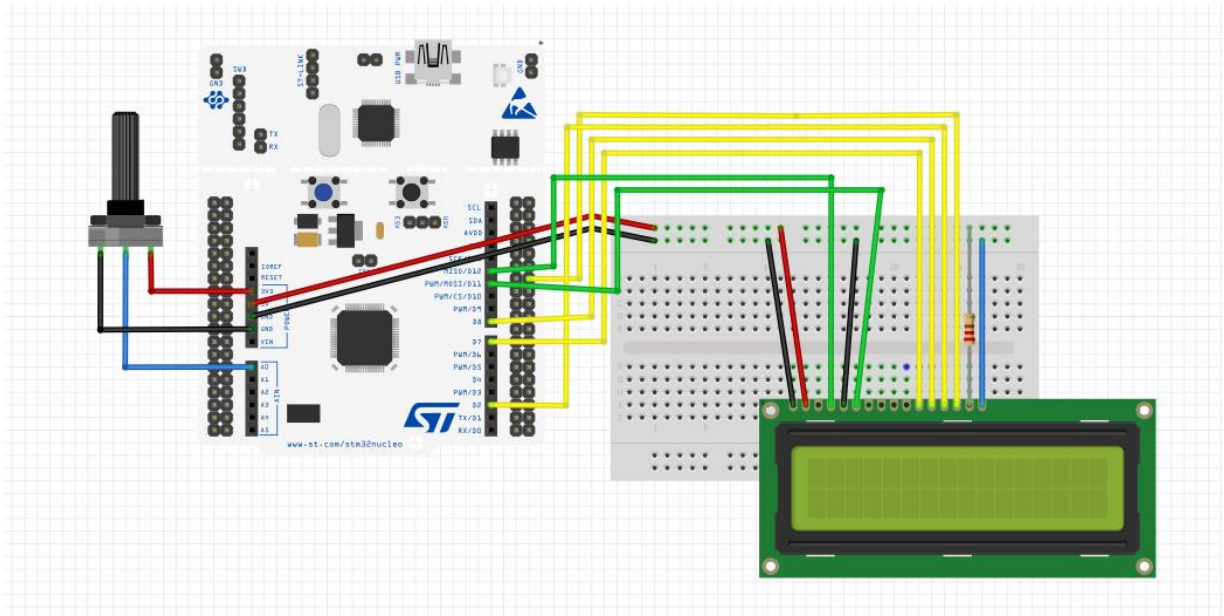
Görüldüğü gibi tek bir kanaldan sürekli olarak ölçüm yapılıp ADC çalışmaya devam etmektedir. Bu sürekli ölçümü boş yere yapmanın bir anlamı yoktur elbette. **DMA** yani doğrudan bellek erişimi ve kesmeler ile kullanıldığında oldukça hızlı ve verimli bir kullanım olur. Bu sürekli ölçüm arka planda olduğundan sürekli güncel veri depolanmaktadır. O yüzden değer / zaman olarak ölçüm yapacaksak bunu kullanmamız gereklidir.



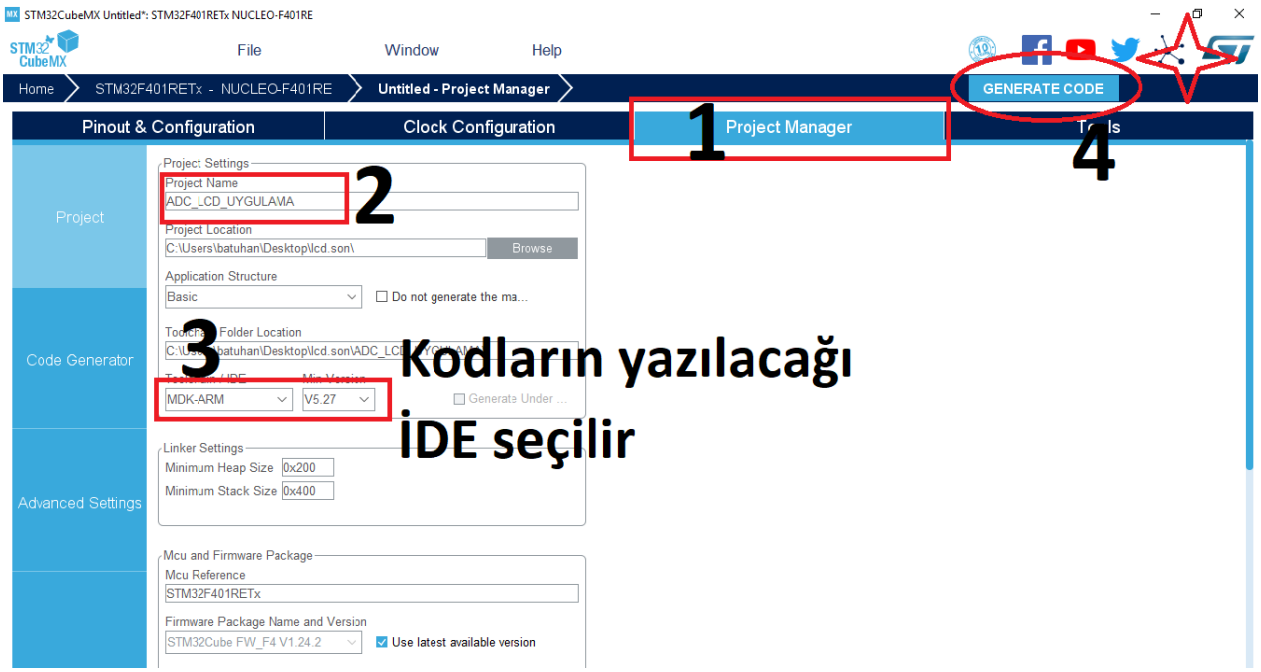
ADC1 kanalın **IN0** girişi seçildiğinden **PA0** pini CUBEMX tarafından ADC okuması için atanır.

LCD nin RS,E,D4,D5,D6,D7 pinleri için **PA6,PA7,PA8,PA9,PA10,PA11** pinleri çıkış olarak ayarlanır.





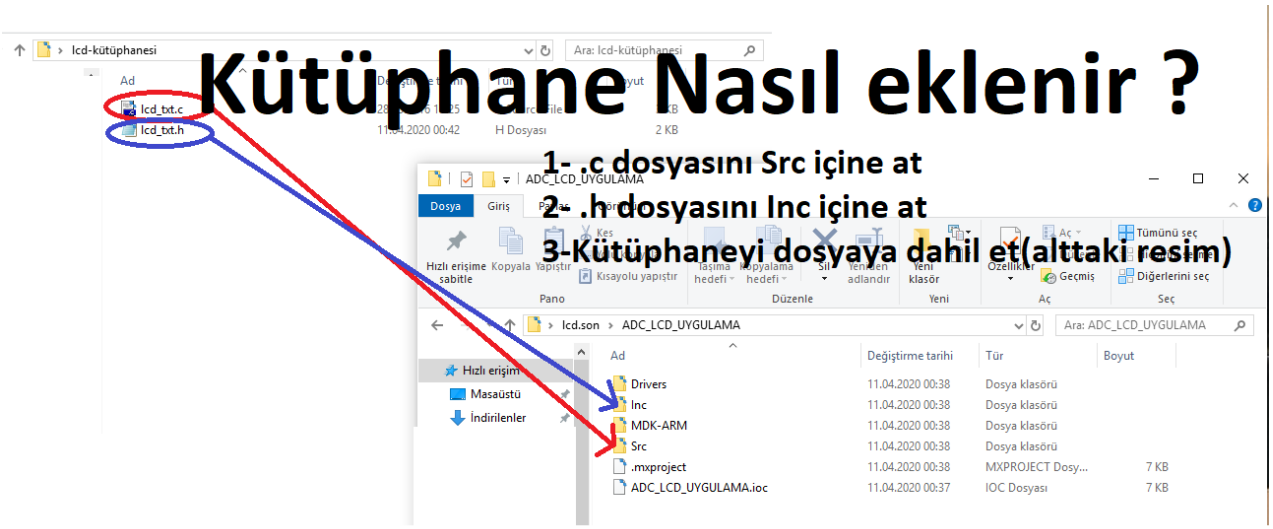
LCD ve Potansiyometre bağlantıları yukarıda verilmiştir.



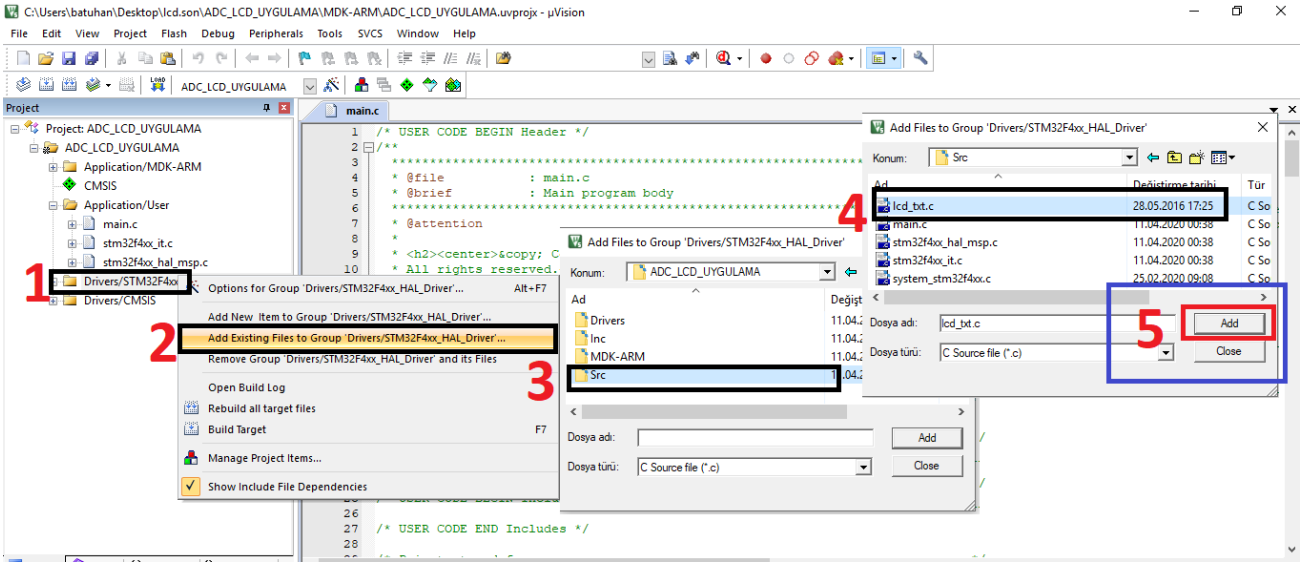
Giriş ve çıkış pinleri ayarlandıktan sonra **Project Manage**r sekmesinden projeye isim verilir.İDE seçilir ve **Generate Code** botununa basılır.

KEİL UVİSİON 5 'DE KODLARIN YAZILMASI

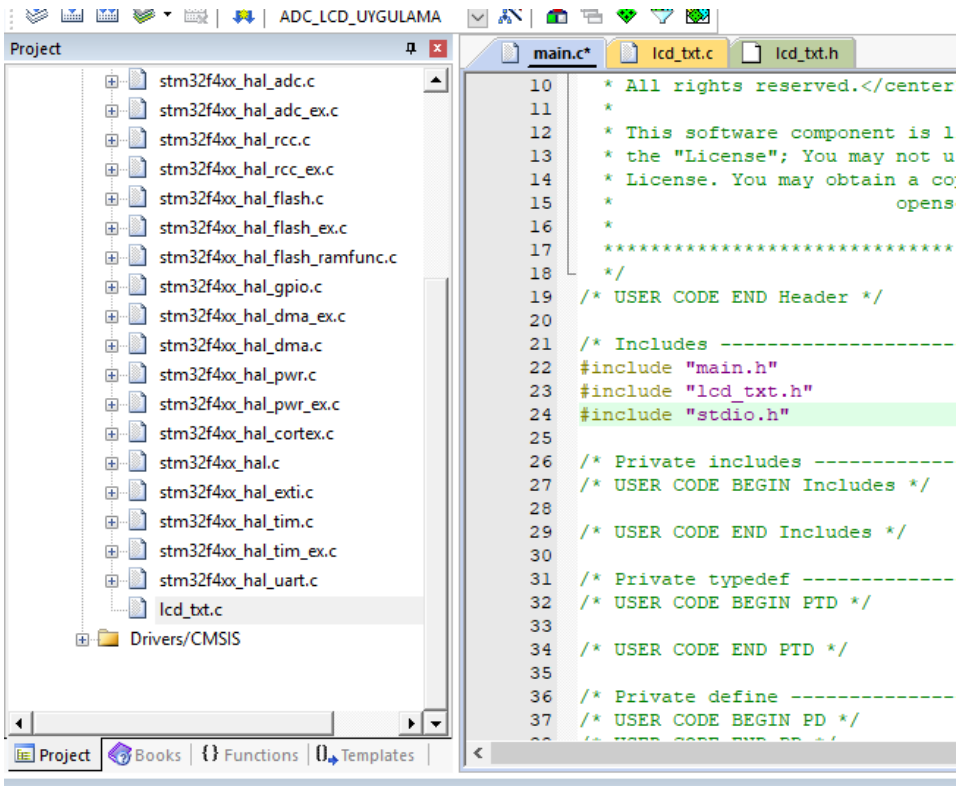
LCD Kütüphanesinin Eklenmesi



LCD kütüphanesinin **.c uzantılı dosyası**, CUBEMX tarafından oluşturulan projenin içindeki **Src** klasörünün içine atılır. **H. Uzantılı dosyası** da **Inc** klasörünün içine atılır.



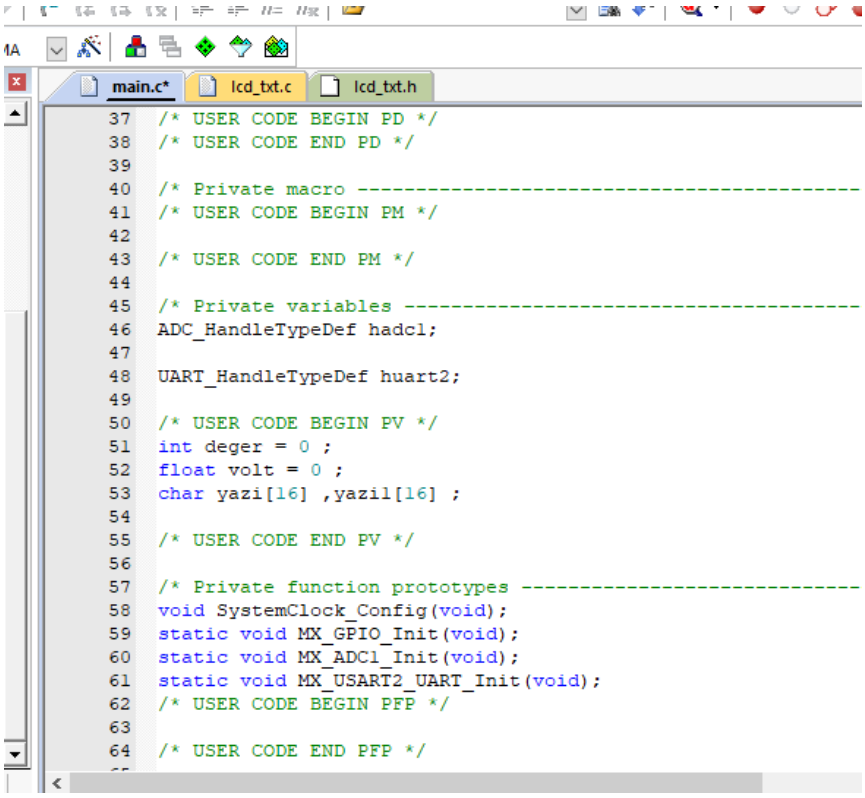
Src klasörünün içine atılan .c uzantılı kütüphane dosyası seçilerek projeye eklenir.



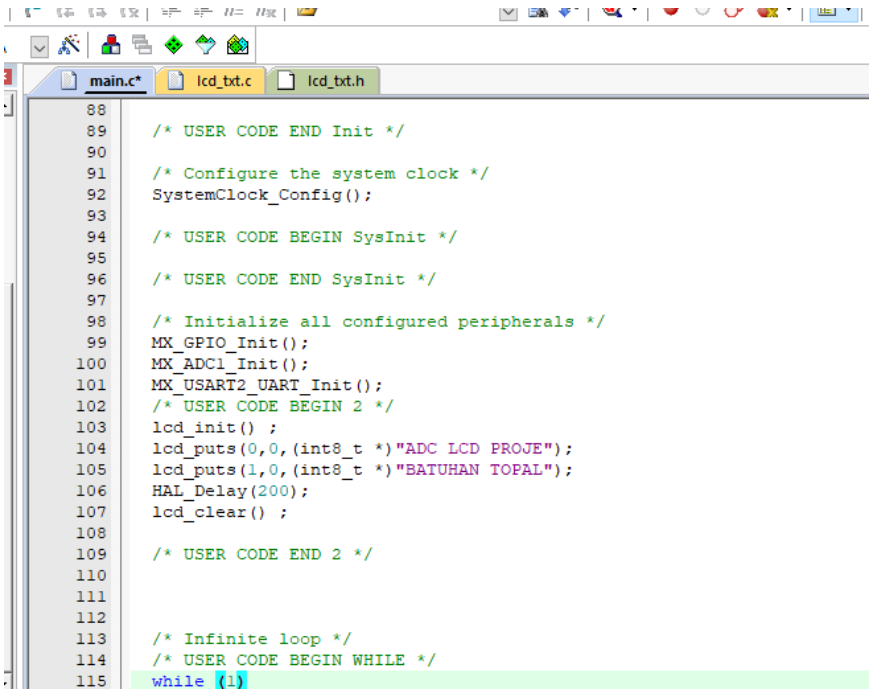
Lcd kütüphanesi main.c nin içine eklenir.

Stdio.h kütüphanesinin eklenmesinin nedeni içinde **sprintf** komutunu barındırmasıdır.

sprintf() fonksiyonu, string (dizgi) içerisine yazdırmak için kullanılır.



Int tipinde deger adında bir degısken tanımlandı.Adc ile okunan değeri volta çevirmek için **float tipinde bir volt** değişkeni tanımlandı.Bu değerleri LCD ye doğrudan yazdıramadığımız için sprintf() komutu için **2 adet yazi[16],yazi1[16] dizi** oluşturuldu.



Lcd_init() fonksiyonu ile LCD hazır hale getirildi.

Program boyunca sadece başlangıçta 1 defaya mahsus “**ADC LCD PROJE”** ve alt satırına **“YUKLENIYOR...”** yazdırıldı ve 200 ms geçtikten sonra ekran temizlendi.

main.c* | lcd_txt.c | lcd_txt.h

```
110
111
112
113   /* Infinite loop */
114   /* USER CODE BEGIN WHILE */
115   while (1)
116   {
117     /* USER CODE END WHILE */
118
119     /* USER CODE BEGIN 3 */
120     HAL_ADC_Start(&hadc1) ;
121     HAL_ADC_PollForConversion(&hadc1,1000);
122     deger=HAL_ADC_GetValue(&hadc1) ;
123     volt= 3.3 * (deger /4095 ) ;
124
125     sprintf(yazi,"ADC:%d  ",deger);
126     lcd_puts(0,0,(int8_t *)yazi) ;
127
128     sprintf(yazi,"VOLT:%f  ",volt);
129     lcd_puts(1,0,(int8_t *)yazil) ;
130
131 }
132   /* USER CODE END 3 */
133 }
134
135 /**
136   * @brief System Clock Configuration
137   * @retval None
```

While(1) sonsuz döngüsünün içine girildiğinde ADC kullanımı başlatılır.

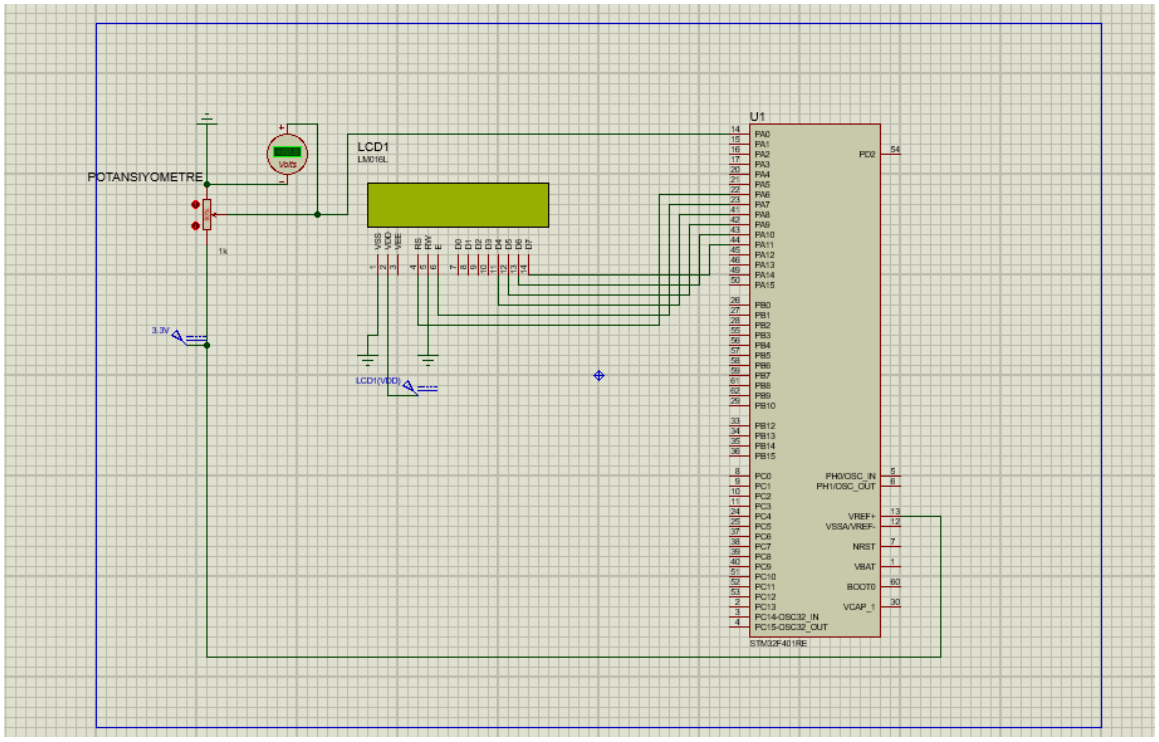
Polling modile sürekli adc okuma yapılır.

GetValue komutu ile Adc okunur ve deger degiskeninin içine atanır.

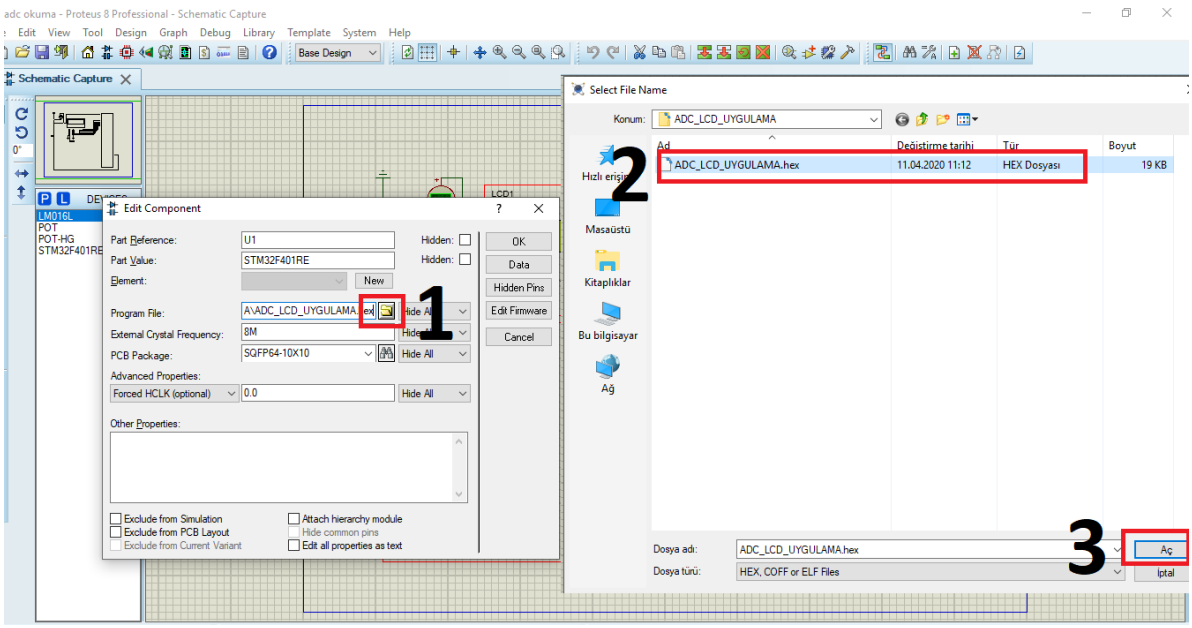
0-4095 arasında değer alan adc'yi volta çevrilir.Ornegin 3000 olan değerin volt karşılıgı 2.417 Volttur.

Her iki değerde LCD ekrana yazdırılır.

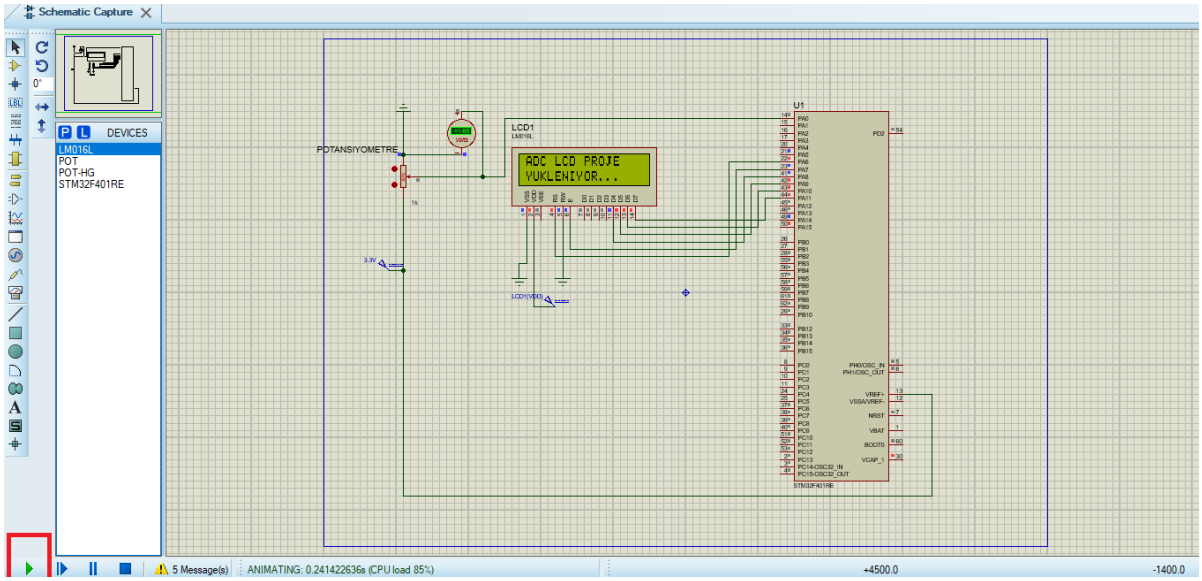
PROJENİN PROTEUS DA ÇALIŞTIRILMASI



LCD bağlantıları CUBEMX de kulanıcı tarafından seçilen pinlerdir .Potansiyometre üzerindeki voltu ölçmek için paralel bir **DC Voltmetre** bağlanır.

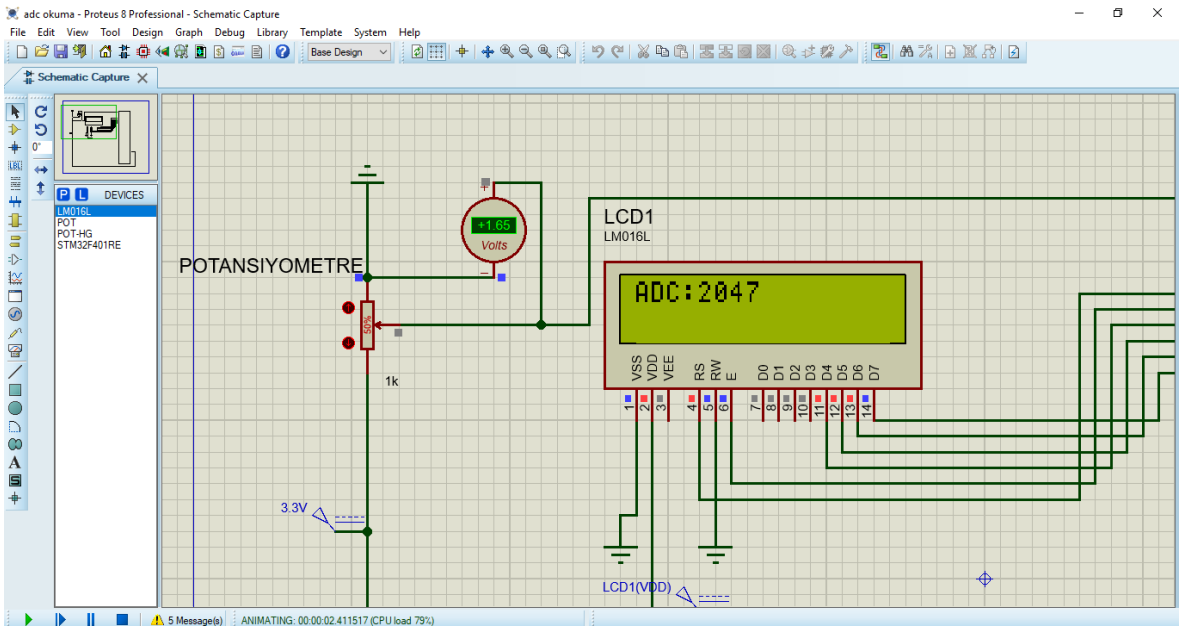


Proteus'da işlemciye kod yüklemek için işlemcinin üzerine tıklanır.Açılan pencerede yukarıdaki **1** numaralı yere basılarak **hex** dosyası seçilir ve **Aç'** a basılarak kod yüklenir.



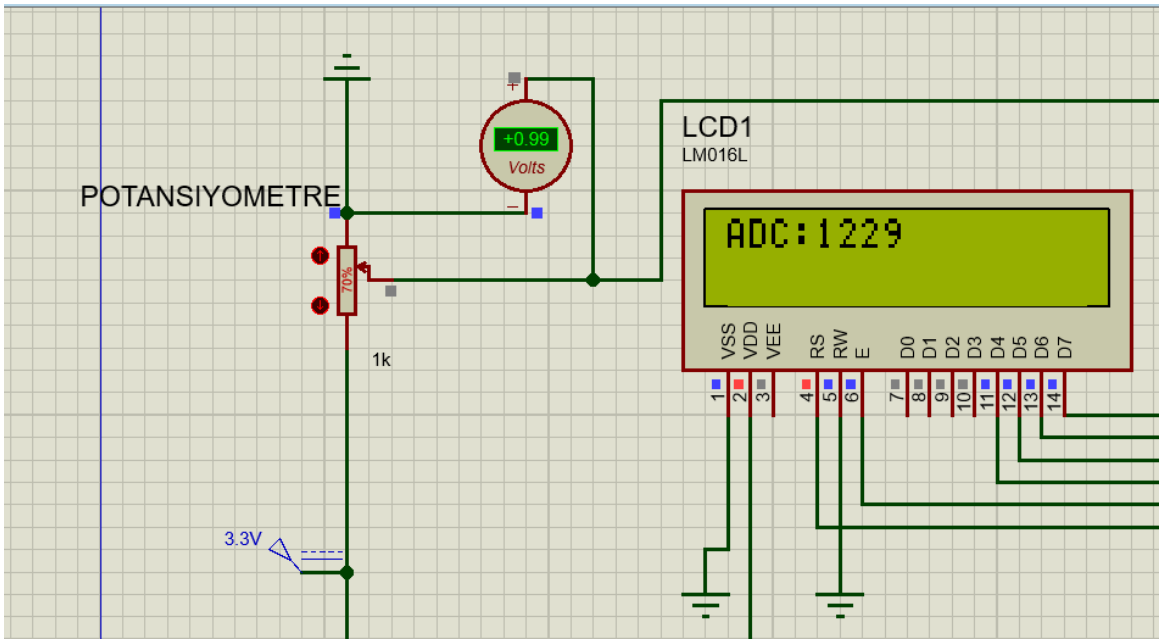
Sol altta bulunan Başlat simgesine basılır ve proje çalışmaya başlar.

Main() in altında (while(1) in değil) yazılan **“ADC LCD PROJE”** ve **“YUKLENIYOR...”** yazıları 1 defa ekrana yazdırılır ve sonrasında silinir.



Referans voltajı olarak 3.3 V ayarlanır..Nedeni STM32F4 serilerinde ADC okumasının maxımum hızda çalışmasını sağlamaktır.

Potansiyometre üzerindeki volt, voltmetre ile ölçülür.

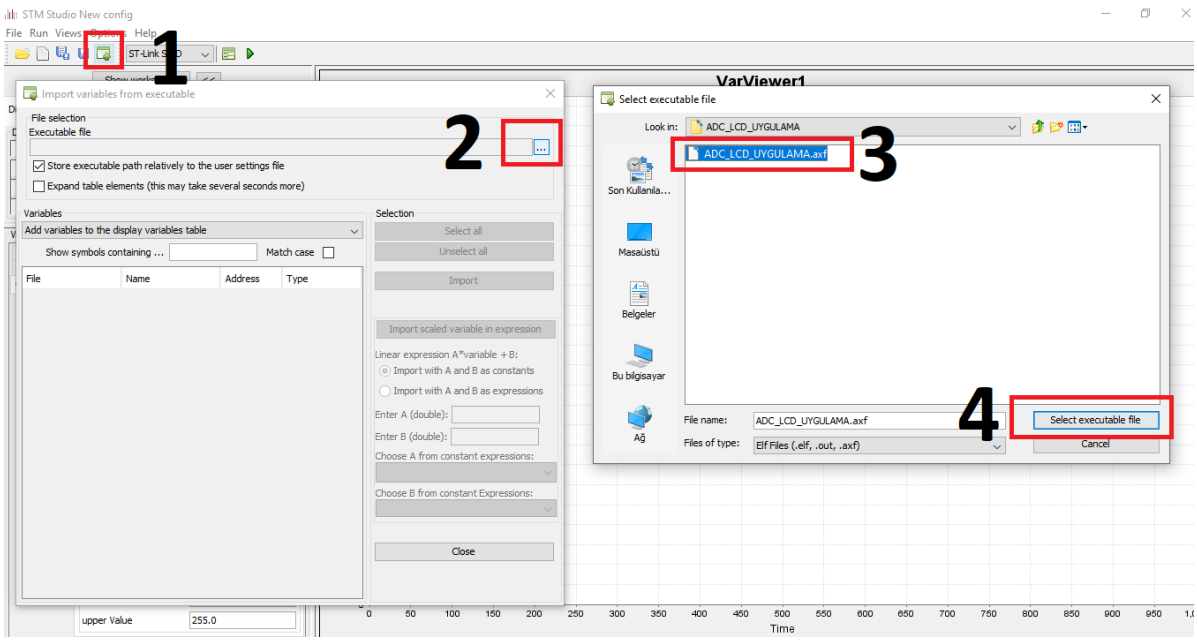


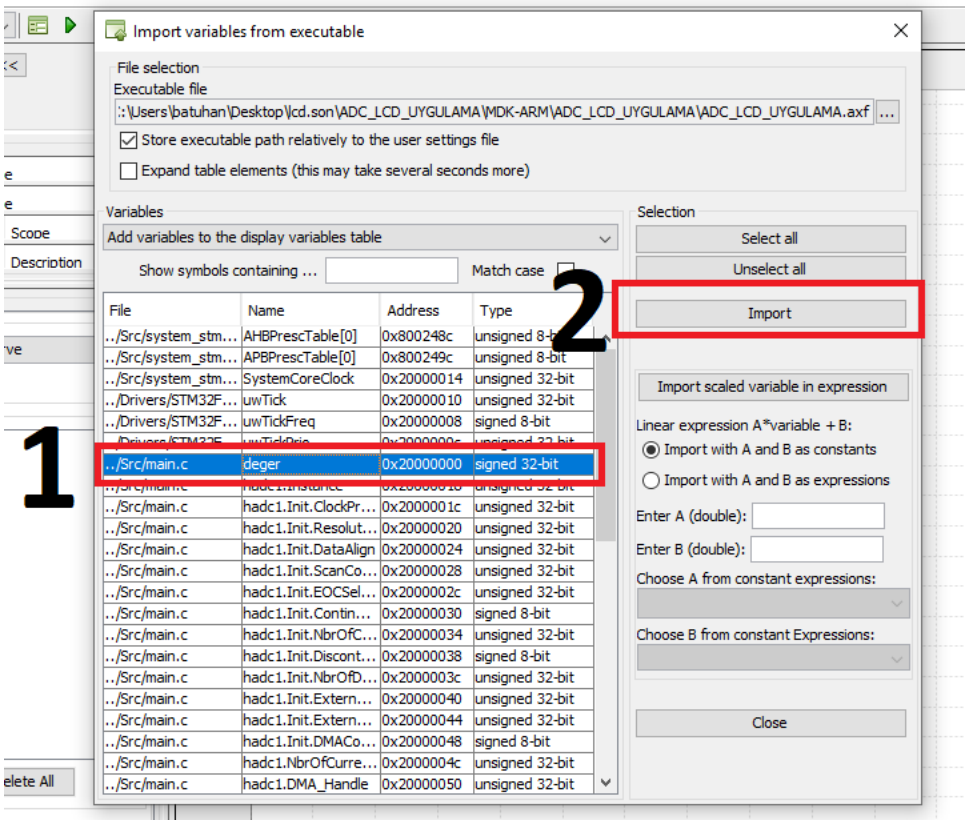
Potansiyometrenin değeri manuel olarak değiştirildiğinde ADC okuması **Countinous Conversion Mode** Aktif olduğundan adc değeri de değişir.

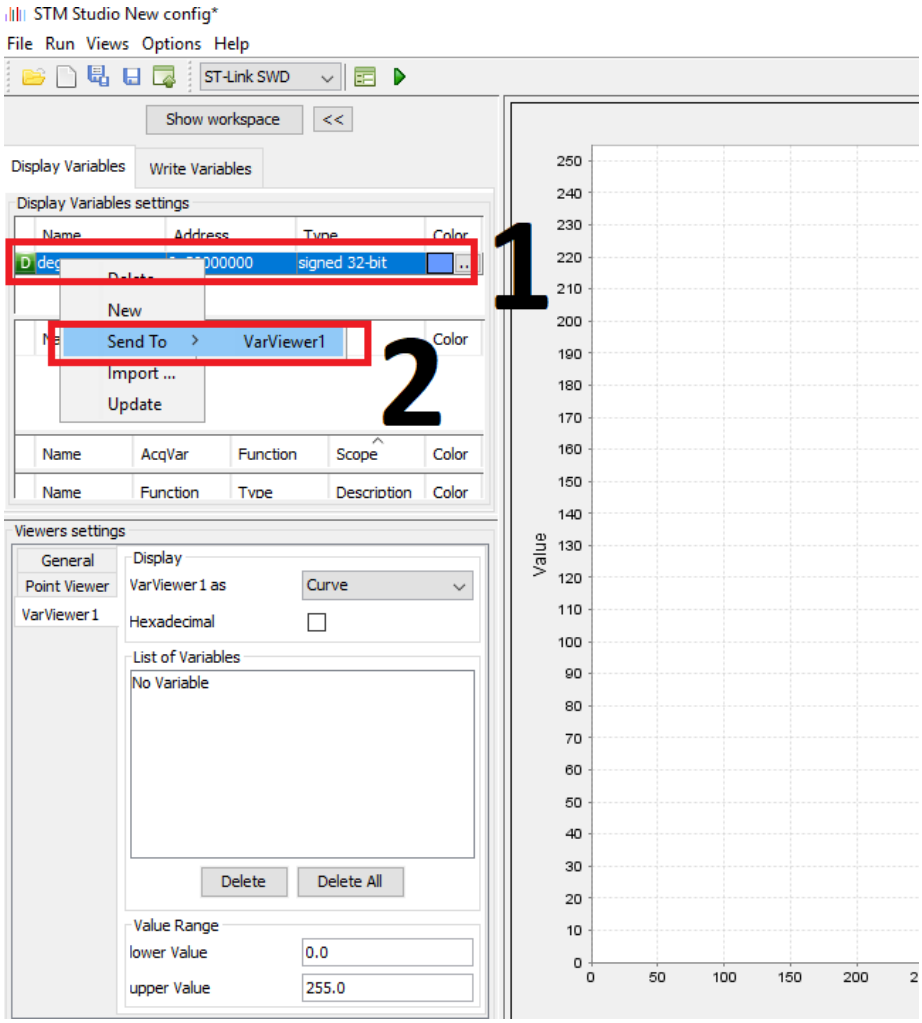
STM STUDİO KULLANIMI

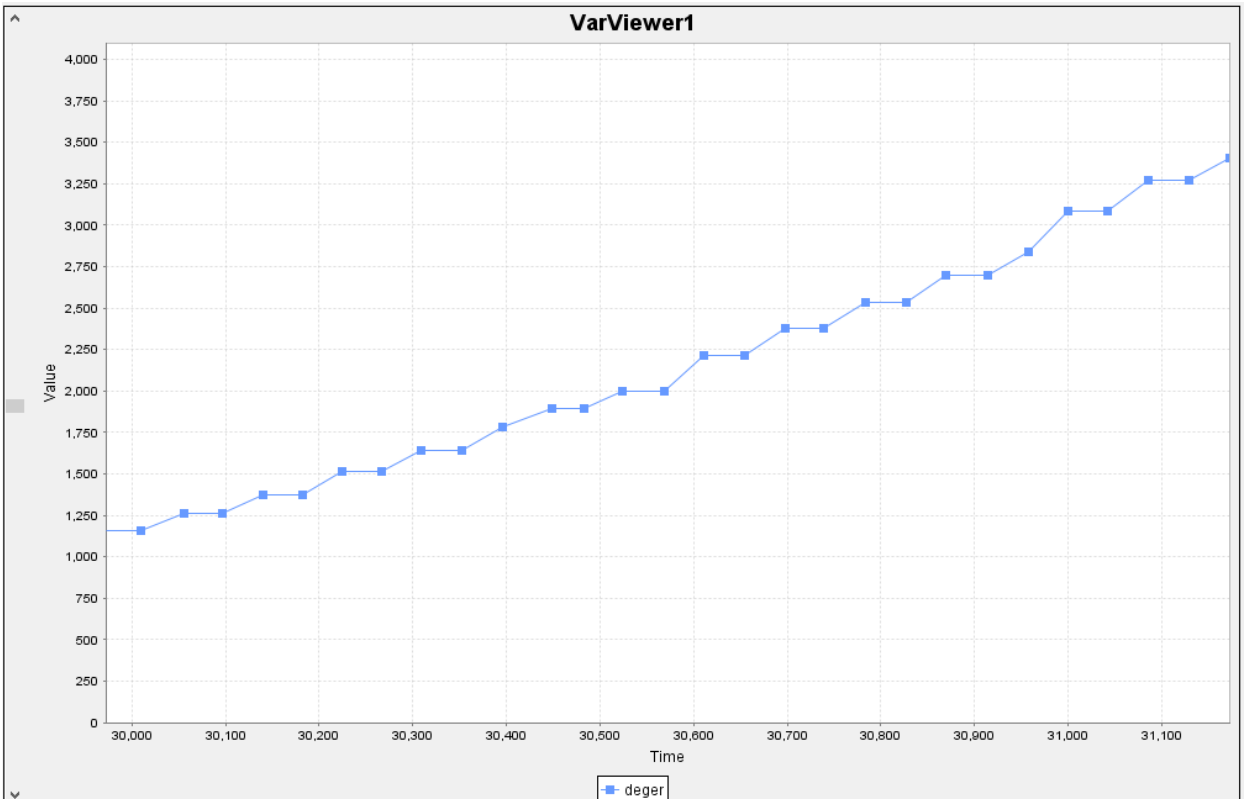
STM Studio mikro denetleyicinin çalışma esnasında değişkenlerin görsel olarak izlenebilmesini sağlayan bir araçtır.

Potansiyometrenin değerini anlık olarak grafik şeklinde incelenmesi için aşağıdaki adımlar sırasıyla yapılır.

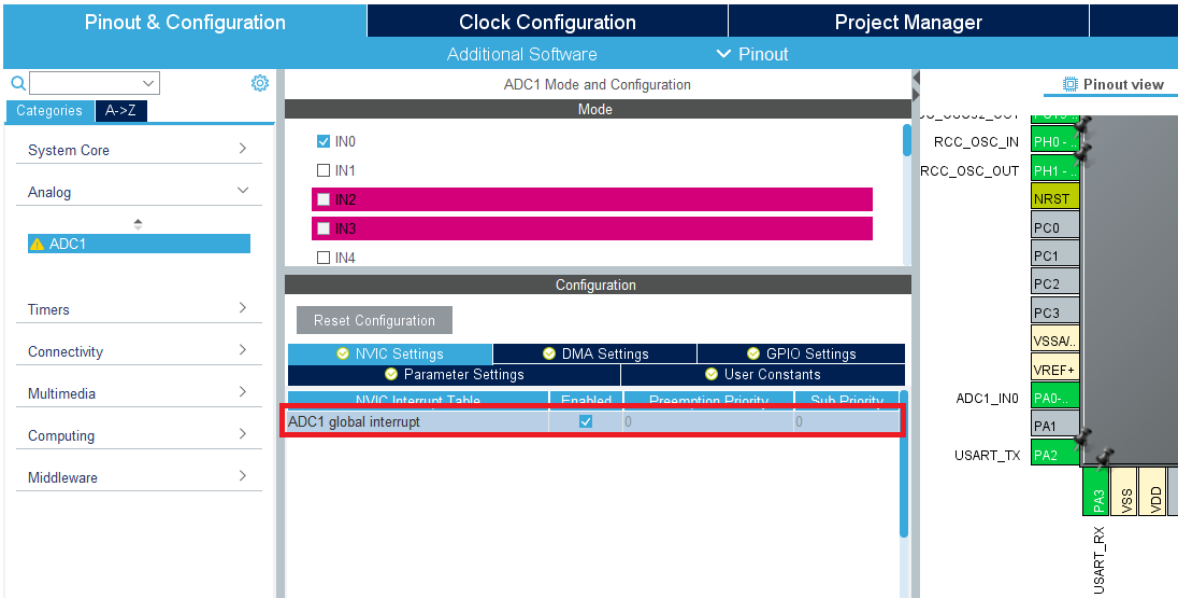



Import variables from executable
File selection
Executable file
::\Users\batuhan\Desktop\cd.son\ADC_LCD_UYGULAMA\MDK-ARM\ADC_LCD_UYGULAMA\ADC_LCD_UYGULAMA.axf
Store executable path relatively to the user settings file
Expand table elements (this may take several seconds more)
Variables
Add variables to the display variables table
Show symbols containing ...
Match case
File
Name
Address
Type
../Src/main.c
deger
0x20000000
signed 32-bit
Selection
Select all
Unselect all
Import
Import scaled variable in expression
Linear expression A*variable + B:
Import with A and B as constants
Import with A and B as expressions
Enter A (double):
Enter B (double):
Choose A from constant expressions:
Choose B from constant Expressions:
Close
1
2


STM Studio New config*
File Run Views Options Help
ST-Link SWD
Show workspace
<<
Display Variables
Write Variables
Display Variables settings
Name
Address
Type
Color
signed 32-bit
New
Send To
VarViewer1
Import ...
Update
Name
AcqVar
Function
Scope
Color
Viewers settings
General
Point Viewer
VarViewer 1
Display
VarViewer 1 as
Curve
Hexadecimal
List of Variables
No Variable
Delete
Delete All
Value Range
lower Value
0.0
upper Value
255.0
Value
1
2



INTERRUPT KULLANIMI



Interrupt kullanılması için **NVIC** ayarlarından **ADC1 Global interrupt** aktif edilir.

```
61
62  /* USER CODE END PFP */
63
64  /* Private user code ---------------------------------------------------------
65  /* USER CODE BEGIN 0 */
66  int deger = 0 ;
67  float volt = 0 ;
68  char yazi[16] ,yazil[16] ;
69
70  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
71    if (hadc == &hadc1) {
72    deger = HAL_ADC_GetValue(&hadc1) ;
73    }
74  }
75
76  /* USER CODE END 0 */
77
```

Interrupt ile kullanırken ADC çevrim işlemini arka planda yapar ve bittiği zaman bir interrupt oluşturur. Böylece dönüştürme işlemi bittiğini anlayıp değeri ADC' den istenilebilir.

Polling modundan farkı ADC çevriminin arka planda yapılmasıdır.

HAL_ADC_ConvCpltCallback() fonksiyonu ile kullanılır.

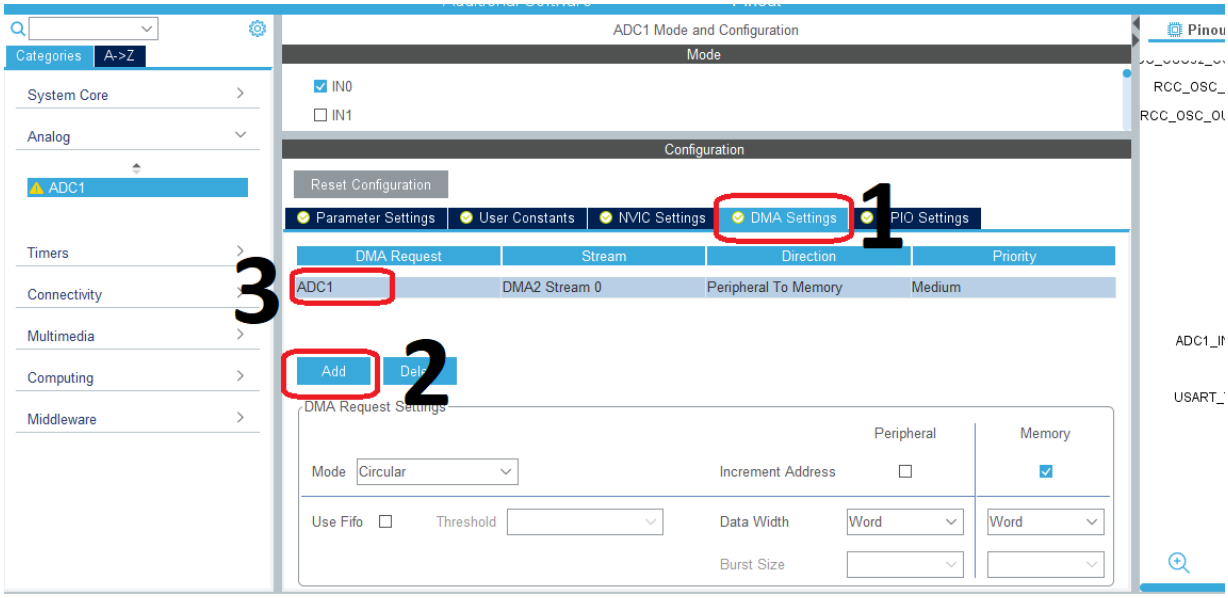
Interrupt callback fonksiyonunu yukarıdaki gibi tanımlanır. Burada önemli olan nokta bu fonksiyonu herhangi yerden **yazılımsal olarak çağırılmadığıdır**. ADC çevrimi tamamlandığında bu fonksiyon çalışmaya başlayacaktır.

```
    MX_ADC1_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    lcd_init() ;
    lcd_puts(0,0,(int8_t *)"ADC LCD PROJE");
    lcd_puts(1,0,(int8_t *)"INTERRUPT YONTEMI");
    HAL_Delay(200);
    lcd_clear() ;

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */
      HAL_ADC_Start_IT(&hadc1);

      sprintf(yazi,"ADC:%d  ",deger);
      lcd_puts(0,0,(int8_t *)yazi) ;

      sprintf(yazil,"VOLT:%f  ",volt);
      lcd_puts(1,0,(int8_t *)yazil) ;

  }
    /* USER CODE END 3 */
  }
```

DMA KULLANIMI

DMA yani doğrudan bellek erişimi ve kesmeler ile kullanıldığında oldukça hızlı ve verimli bir kullanım olur. Bu sürekli ölçüm arka planda olduğundan sürekli güncel veri depolanmaktadır. O yüzden değer/zaman olarak ölçüm yapacaksak bunu kullanmamız gereklidir.

DMA , adc sayısının çok olması durumda çevrimi kolaylaştıran bir yöntemdir.




```
63
64  /* USER CODE END PFP */
65
66  /* Private user code ---------------------------------------------
67  /* USER CODE BEGIN 0 */
68  int deger = 0 ;
69  float volt = 0 ;
70  uint32_t adcbuffer[1] ;
71  char yazi[16] ,yazil[16] ;
72
73  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
74  if (hadc == &hadc1) {
75   deger = adcbuffer[0] ;
76   }
77   }
78
79  /* USER CODE END 0 */
80
81 /**
82    * @brief  The application entry point.
83    * @retval int
84    */
```

DMA kullanarak adc okuması yapıldığında **ADC_Get_Value()** Komutuna ihtiyaç duyulmaz.

```
104   /* USER CODE BEGIN SysInit */
105
106   /* USER CODE END SysInit */
107
108   /* Initialize all configured peripherals */
109   MX_GPIO_Init();
110   MX_DMA_Init();
111   MX_ADC1_Init();
112   MX_USART2_UART_Init();
113   /* USER CODE BEGIN 2 */
114
115   lcd_init() ;
116   lcd_puts(0,0,(int8_t *)"ADC LCD PROJE");
117   lcd_puts(1,0,(int8_t *)"DMA YONTEMI");
118   HAL_Delay(200);
119   lcd_clear() ;
120
121   HAL_ADC_Start_DMA(&hadc1,adcbuffer,1) ;
122
123
124   /* USER CODE END 2 */
```

Main() in altında DMA başlatılır.ADC çevrimi arka planda yapılır. **Adcbuffer[1]** ın içine kaybedilir.

While(1) in içinde değer lazım olduğunda bellekten çekilir ve kullanılır