



Teknoloji Fakültesi Elektrik Elektronik Mühendisliği Bölümü

EE-302

Mikroişlemciler

Analog/Dijital Çevrim Modülü
ADC Uygulamaları

9. Hafta

Prof. Dr. Mehmet DEMİRTAŞ

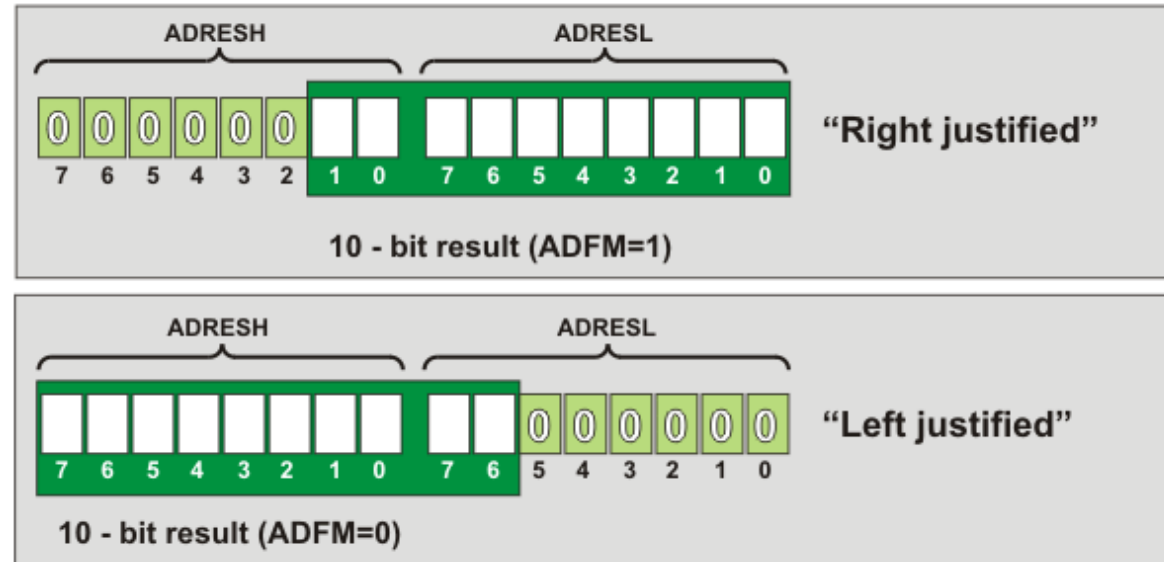
Analog/Dijital (ADC modülü)

- Doğada doğal olarak bulunan tüm sistemler analog düzendedir. Kullandığımız mikrodenetleyiciler ise dijital sistemlerdir. Bu nedenle dış dünya ile iletişim kurmak ve dış dünyadan veri almak - algılama yapmak - için bu iki sistemin bir şekilde birbirini anlaması gereklidir. Bu iş içinde doğada bulunan analog sinyallerin (ısı, ışık, ses, nem vb.) dijital sistemlerin anlayacağı dijital sinyallere çevrilmesi gereklidir. Bu işlem için ADC devreleri ve entegreleri kullanılmaktadır. PIC16F877 mikrodenetleyicisinde ADC modülü bulunmaktadır. Böylece harici bir devre veya entegre kullanmadan analog sinyaller dahili ADC modülü sayesinde dijital sinyallere çevrilebilir. Şekil 1'de PIC16F877'de bulunan ADC modülünün blok diyagramı görülmektedir.



Analog/Dijital (ADC modülü)

- PIC16F877 8 ADC giriş ucu (RA0/AN0, RA1/AN1, RA2/AN2, RA3/AN3, RA5/AN4, RE0/AN5, RE1/AN6, RE2/AN7) vardır. ADC modülünün çözünürlüğü 10 bit'tir. ADC birimi için gerekli referans voltajı VDD, Vss, RA2 ve RA3 uçları ve/veya bunların kombinasyonlarından seçilerek elde edilir.
- ADC birimi 4 adet kaydediciye sahiptir. ADC çevrim sonucunun yazıldığı ADRESH (ADRES High Register), ADRESL (ADRES Low Register). ADC birimi kontrol kaydedicileri ise ADCON0 (A/D Control Register0) ve ADCON1'dir (A/D Control Register1).



ADCON0 Register

| ADCON0 | R/W (0) | R/W (0) | R/W (0) | R/W (0) | R/W (0) | R/W (0) | (0) | R/W (0) | Features |
|--------|---------|---------|---------|---------|---------|---------|-------|---------|----------|
| | ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/DONE | - | ADON | Bit name |
| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

Legend

R/W Readable/Writable bit
(0) After reset, bit is cleared

Bit 7-6 **ADCS1:ADCS0** = A/D Çevrim Clock Seçme Bit'i.

00 = $F_{OSC}/2$

01 = $F_{OSC}/8$

10 = $F_{OSC}/32$

11 = F_{RC} (Dahili RC Osilatör)

Bit 5-3 **CHS2:CHS1** = Analog Sinyal Giriş Kanalı Seçme Bit'i. Seçilen uçtaki sinyalin analog değeri dijital bilgiye çevrilir.

000: Kanal 0, (RA0, AN0)

001: Kanal 1, (RA1, AN1)

010: Kanal 2, (RA2, AN2)

011: Kanal 3, (RA3, AN3)

100: Kanal 4, (RA4, AN4)

101: Kanal 5, (RA5, AN5)

110: Kanal 6, (RA6, AN6)

111: Kanal 7, (RA7, AN7)

ADCON0 Register

| ADCON0 | R/W (0) | R/W (0) | R/W (0) | R/W (0) | R/W (0) | R/W (0) | (0) | R/W (0) | Features |
|--------|---------|---------|---------|---------|---------|---------|-------|---------|----------|
| | ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/DONE | - | ADON | Bit name |
| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

Legend

| | |
|------------|--|
| R/W (0) | Readable/Writable bit After reset, bit is cleared |
|------------|--|

- Bit 2 **GO/DONE** = A/D Çevrim Durum Bit'i
Eğer ADON bit'i 1 ise;
1: A/D modülü işlem yapıyor. (ADC çevrimine başlamak için ayarlanır.)
0: A/D modülü işlem yapmıyor. (Bu bit ADC çevrimi bitince otomatik olarak 0 olur.)
- Bit 1 Kullanılmayan bit. Okunduğunda "0" değeri alınır.
- Bit 0 **ADON** = A/D On Bit'i
1: ADC modülü açık
0: ADC modülü kapalı

ADCON1 Register

| ADCON1 | R/W (0) | | | | R/W (0) | R/W (0) | R/W (0) | R/W (0) | Features |
|--------|---------|-------|-------|-------|---------|---------|---------|---------|----------|
| | ADFM | - | - | - | PCFG3 | PCFG2 | PCFG1 | PCFG0 | Bit name |
| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

Legend

| | |
|---------|--|
| - | Bit is unimplemented |
| R/W (0) | Readable/Writable bit After reset, bit is cleared |

Bit 7 **ADFM**

= A/D çevirimi sonuç Formatı Yazılma Seçenekleri

- 1: ADRESH ve ADRESL kaydedicileri 8'er bit'tir. Toplam 16 bit'lik sayıyı muhafaza edebilirler. ADC çevrim sonucu 10 bit'lik sayı olarak bu kaydedicilere yazılır. Bu bit lojik-1 olunca sonuç **sağa dayalı** olarak yazılır. Yani ADRESH kaydedicisi içindeki 6 yüksek değerlikli bit'in değeri 0'dır. Çünkü sonuç sağa dayalı yazılmıştır.
- 0: ADC çevrim sonucu sola dayalı olarak yazılır. Yani ADRESL kaydedicisi içindeki 6 az değerlikli bit'in değeri 0'dır.

Bit 6-4

= Kullanılmayan bit'ler. Okunduğunda "0" değeri alınır.

ADCON1 Register

| | | | | | | | | | |
|--------|---------|-------|-------|-------|---------|---------|---------|---------|----------|
| ADCON1 | R/W (0) | | | | R/W (0) | R/W (0) | R/W (0) | R/W (0) | Features |
| | ADFM | - | - | - | PCFG3 | PCFG2 | PCFG1 | PCFG0 | Bit name |
| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

Bit 3-0 **PCFG3:PCFG0**

= ADC Analog Uç Ayar Bit'leri. Aşağıdaki tabloda bu bit'lere verilen değerlere göre analog uçların hangilerinin voltaj referans gerilim ucu, hangilerinin analog sinyal giriş ucu olacağı belirlenir.

| PCFG <3:0> | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 | VREF+ | VREF- | C/R |
|---------------|-----|-----|-----|-----|-------|-------|-----|-----|-------|-------|-----|
| 0000 | A | A | A | A | A | A | A | A | VDD | VSS | 8/0 |
| 0001 | A | A | A | A | VREF+ | A | A | A | AN3 | VSS | 7/1 |
| 0010 | D | D | D | A | A | A | A | A | VDD | VSS | 5/0 |
| 0011 | D | D | D | A | VREF+ | A | A | A | AN3 | VSS | 4/1 |
| 0100 | D | D | D | D | A | D | A | A | VDD | VSS | 3/0 |
| 0101 | D | D | D | D | VREF+ | D | A | A | AN3 | VSS | 2/1 |
| 011x | D | D | D | D | D | D | D | D | — | — | 0/0 |
| 1000 | A | A | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 6/2 |
| 1001 | D | D | A | A | A | A | A | A | VDD | VSS | 6/0 |
| 1010 | D | D | A | A | VREF+ | A | A | A | AN3 | VSS | 5/1 |
| 1011 | D | D | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 4/2 |
| 1100 | D | D | D | A | VREF+ | VREF- | A | A | AN3 | AN2 | 3/2 |
| 1101 | D | D | D | D | VREF+ | VREF- | A | A | AN3 | AN2 | 2/2 |
| 1110 | D | D | D | D | D | D | D | A | VDD | VSS | 1/0 |
| 1111 | D | D | D | D | VREF+ | VREF- | D | A | AN3 | AN2 | 1/2 |

A = Analog input D = Digital I/O

C/R = # of analog input channels/# of A/D voltage references

ADCON1 Register

| PCFG <3:0> | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 | VREF+ | VREF- | C/R |
|---------------|-----|-----|-----|-----|-------|-------|-----|-----|-------|-------|-----|
| 0000 | A | A | A | A | A | A | A | A | VDD | VSS | 8/0 |
| 0001 | A | A | A | A | VREF+ | A | A | A | AN3 | VSS | 7/1 |
| 0010 | D | D | D | A | A | A | A | A | VDD | VSS | 5/0 |
| 0011 | D | D | D | A | VREF+ | A | A | A | AN3 | VSS | 4/1 |
| 0100 | D | D | D | D | A | D | A | A | VDD | VSS | 3/0 |
| 0101 | D | D | D | D | VREF+ | D | A | A | AN3 | VSS | 2/1 |
| 011x | D | D | D | D | D | D | D | D | — | — | 0/0 |
| 1000 | A | A | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 6/2 |
| 1001 | D | D | A | A | A | A | A | A | VDD | VSS | 6/0 |
| 1010 | D | D | A | A | VREF+ | A | A | A | AN3 | VSS | 5/1 |
| 1011 | D | D | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 4/2 |
| 1100 | D | D | D | A | VREF+ | VREF- | A | A | AN3 | AN2 | 3/2 |
| 1101 | D | D | D | D | VREF+ | VREF- | A | A | AN3 | AN2 | 2/2 |
| 1110 | D | D | D | D | D | D | D | A | VDD | VSS | 1/0 |
| 1111 | D | D | D | D | VREF+ | VREF- | D | A | AN3 | AN2 | 1/2 |

A = Analog input D = Digital I/O

C/R = # of analog input channels/# of A/D voltage references

Analog/Dijital (ADC modülü)

- 10 bit'lik bir ADC en fazla $2^{10} = 1024$ adet değer ile bir analog işareti örnekleyebilir. ADC biriminin elde ettiği dijital bilginin bit sayısı ADC modülünün çözünürlüğünü ifade eder. Çözünürlük ne kadar yüksekse o kadar iyi bir dönüşüm yapılır. Örneğin; ADC'nin dijital bilgiye çevireceği sinyalin maksimum gerilim değeri yani belirlenen V_{REF} değeri 5V ise ADC **adım büyüklüğü** şu şekilde hesaplanır.

$$\text{Adım Büyüklüğü} = \frac{\text{Sinyalin Maksimum Gerilim Değeri}}{2^{\text{ADC Bit Sayısı}}} = \frac{5V}{2^{10}} = \frac{5}{1024} = 0,0048828125V$$

- Dijital çıkış değerlerini (10 bit'lik ADC için) Tablo'da daha iyi görebilirsiniz. 10 bit'lik ADC için, dijital değer 0000000000 (desimal 0)'dan başlar 1111111111 (desimal 1023)'e kadar gider. 0'dan başladığı için son değer 1024 değil 1023 olur.

Analog/Dijital (ADC modülü)

| Analog Giriş (V) | Sayısal Bilgi Karşılığı |
|------------------|-------------------------|
| 0,00 | 0000000000 |
| 0,0048828125 | 0000000001 |
| 0,009765625 | 0000000010 |
| 0,146484375 | 0000000011 |
| 0,01953125 | 0000000100 |
| 0,0244140625 | 0000000101 |
| ... | ... |
| ... | ... |
| ... | ... |
| 4,9951171875 | 1111111111 |

- ADC biriminin analog uçlarına gelen sinyalin gerilim değerini hesaplamak içinse aşağıdaki formül kullanılır.
- ADC Girişine Gelen Sinyal Gerilimi=Sayısal Çıkış Değerinin Desimal Değeri x Adım Büyüklüğü
- Örnek olarak 10 bit'lik ADC ucuna gelen analog sinyalin dijital karşılığı çevrim sonucunda 1000000000 olarak bulunduysa. Bu dijital bilginin gerilim değeri verilen formüle göre aşağıda gösterildiği gibi hesaplanır.
- ADC Girişine Gelen Sinyalin Gerilimi=512 (binary olarak 1000000000)x0,0048828125= 2,5V

- **A/D Kesmesi**

- ADC birimi için aynı zamanda birde kesme mevcuttur. ADC işlemi bitince istenirse kesme meydana getirtilebilir. #INT_AD kesmesi, ADC birimi analog sinyali dijital bilgiye çevirme işlemini bitirince meydana gelir.

- **SETUP_ADC () FONKSİYONU**

- Bu fonksiyon ile A/D biriminin mod ayarı yapılır. ADC'nin kapalı olup olmayacağı, ADC frekansı olarak mikrodenetleyicinin kendi frekansını mı? yoksa dışarıdan bir giriş frekansını mı? kullanacağını, ADC işlemi için denetleyicinin kendi frekansını kullanacaksa bu frekans kaç bölerek kullanacağı gibi işlemleri belirlemek için kullanılır. Kullanılacak "mod " seçenekleri mikrodenetleyici modeline göre değişebilir. Kullanılan mikrodenetleyicinin tanıtım dosyasında (16f877.h gibi) bu mod seçeneklerine bakılarak kullanılabilecek komutlar öğrenilir.

- `setup_adc(mod);`
- `setup_adc(ADC_OFF);` // ADC biriminin kullanılmayacağını belirtir.
- `setup_adc(ADC_CLOCK_INTERNAL);` // ADC çalışma frekansı mikrodenetleyicinin içindeki RC osilatörden olacağını belirtir.
- `setup_adc(ADC_CLOCK_DIV_2);` //ADC çalışma frekansı $f_{ADC} = f_{osc} / 2$
- `setup_adc(ADC_CLOCK_DIV_8);` //ADC çalışma frekansı $f_{ADC} = f_{osc} / 8$
- `setup_adc(ADC_CLOCK_DIV_32);` //ADC çalışma frekansı $f_{ADC} = f_{osc} / 32$

- **SETUP_ADC_PORTS ()**

- ADC birimine sahip PIC mikrodenetleyicilerinde sadece A/D çevrimi için kullanılan özel giriş port'ları yoktur. Genel giriş/çıkış port'ları hem normal kullanım hem de A/D çevrimi için kullanılır. Fakat A/D çevrimi için denetleyicide hangi pin'lerin kullanılacağı bellidir. Programcının yapması gereken bu pin'lerden hangisinin A/D çevriminde analog giriş için kullanılacağı hangisinin de normal dijital giriş/çıkış işlemi için kullanılacağını derleyiciye bildirmesidir. Bu fonksiyon ile kullanılacak analog giriş pin'leri derleyiciye bildirilir. Aynı zamanda bu komut ile A/D çevriminde kullanılacak V_{REF} voltaj girişi belirlenir.
- **setup_adc_ports (sabit tanım);**

- **SETUP_ADC_PORTS ()**
- **ALL_ANALOG** = Denetleyicide A/D çevrimi için ayrılan tüm pin'lerin analog giriş için kullanılacağını ve V_{DD} beslemesinin $+V_{REF}$ olarak, $-V_{REF}$ olarak da V_{SS} beslemesinin kullanılacağını belirtir.
- **NO_ANALOGS** = Denetleyicide A/D çevrimi için ayrılan hiçbir pin'in analog giriş için kullanılmayacağını belirtir.
- **AN0_AN1_AN2_AN4_AN5_AN6_AN7 _VSS_VREF** = RA0, RA1, RA2, RA5, RE0, RE1, RE2 girişlerinin analog giriş olacağını, $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin, $-V_{REF}$ olarak da V_{SS} beslemesinin kullanılacağını belirtir.
- **AN0_AN1_AN2_AN3_AN4** = RA0, RA1, RA2, RA3, RA5 girişlerinin analog giriş olacağını ve V_{DD} beslemesinin $+V_{REF}$ olarak, $-V_{REF}$ olarak da V_{SS} beslemesinin kullanılacağını belirtir.

- **SETUP_ADC_PORTS ()**
- **AN0_AN1_AN2_AN4_VSS_VREF** = RA0, RA1, RA2, RA5 girişlerinin analog giriş olacağını ve RA3 pin'inden alınan sinyalin $+V_{REF}$ olarak kullanılacağını, $-V_{REF}$ olarak da Vss beslemesinin kullanılacağını belirtir.
- **AN0_AN1_AN3** = RA0, RA1, RA3 girişlerinin analog giriş olacağını ve V_{DD} beslemesinin $+V_{REF}$ olarak, $-V_{REF}$ olarak da Vss beslemesinin kullanılacağını belirtir.
- **AN0_AN1_VSS_VREF** = RA0, RA1 girişlerinin analog giriş olacağını, $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin, $-V_{REF}$ olarak da Vss beslemesinin kullanılacağını belirtir.
- **AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF** = RA0, RA1, RA5, RE0, RE1, RE2 girişlerinin analog giriş olacağını, $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin, $-V_{REF}$ olarak da RA2 pin'inden alınan sinyalin kullanılacağını belirtir .

- **SETUP_ADC_PORTS ()**
- **AN0_AN1_AN2_AN3_AN4_AN5** = RA0, RA1, RA2, RA3, RA5, RE0 girişlerinin analog giriş olacağını ve V_{DD} beslemesinin $+V_{REF}$ olarak, $-V_{REF}$ olarak da Vss beslemesinin kullanılacağını belirtir.
- **AN0_AN1_AN2_AN4_AN5_VSS_VREF** = RA0, RA1, RA2, RA5, RE0 girişlerinin analog giriş olacağını ve RA3 pin'inden alınan sinyalin $+V_{REF}$ olarak kullanılacağını $-V_{REF}$ olarak da Vss beslemesinin kullanılacağını belirtir.
- **AN0_AN1_AN4_AN5_VREF_VREF** = RA0, RA1, RA5, RE0 girişlerinin analog giriş olacağını $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin $-V_{REF}$ olarak da RA2 pin'inden alınan sinyalin kullanılacağını belirtir.
- **AN0_AN1_AN4_VREF_VREF** = RA0, RA1, RA5 girişlerinin analog giriş olacağını $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin $-V_{REF}$ olarak da RA2 pin'inden alınan sinyalin kullanılacağını belirtir.

- **SETUP_ADC_PORTS ()**
- **AN0_AN1_VREF_VREF** = RA0, RA1 girişlerinin analog giriş olacağını $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin, $-V_{REF}$ olarak da RA2 pin'inden alınan sinyalin kullanılacağını belirtir.
- **AN0** = RA0 girişinin analog giriş olacağını ve V_{DD} beslemesinin $+V_{REF}$ olarak, $-V_{REF}$ olarak da V_{SS} beslemesinin kullanılacağını belirtir.
- **AN0_VREF_VREF** = RA0 girişinin analog giriş olacağını $+V_{REF}$ olarak RA3 pin'inden alınan sinyalin $-V_{REF}$ olarak da RA2 pin'inden alınan sinyalin kullanılacağını belirtir.
- `setup_adc_ports(NO_ANALOGS);` // Hiçbir giriş analog değil.
- `setup_adc_ports(AN0_AN1_VSS_VREF);` // RA1, RA2 analog RA3 $+V_{REF}$

• SET_ADC_CHANNEL () FONKSİYONU

- Analog - Dijital çevrim yapılacak kanal seçimini yapan fonksiyondur. Analog sinyali dijital sinyale çevir komutu geldiğinde mikrodenetleyicinin hangi girişteki (kanaldaki) analog sinyali dijital bilgiye çevireceğini bilmesi gerekir.
- `set_adc_channel (kanal);`
- Bu fonksiyonda kanal kısmına 0,1,2, diye kanal numaraları yazılır. Her numara kullanılan mikrodenetleyicinin bilgi kitabında (data sheet) pin diyagramı kısmında (AN0, AN1, AN2, ... diye) görülebilir. Her rakam o kanalın sadece numarasını ifade eder. Yani AN2 kanalını ifade etmek için fonksiyonda kanal kısmına 2 yazılır. Örneğin PIC16F877'nin pin diyagramına baktığımızda;
- 0= RA0/AN0 pin'ini, 1= RA1/AN1 pin'ini, 2= RA2/AN2 pin'ini, 3= RA3/AN3 pin'ini, 4= RA5/AN4 pin'ini, 5= RE0/AN5 pin'ini, 6= RE1/AN6 pin'ini, 7= RE2/AN7 pin'ini ifade eder.
- `set_adc_channel(2);` // AN2 ucundaki analog girişinin dijitalle çevrileceğini bildirir.

- **SET_ADC_CHANNEL () FONKSİYONU**

- Belirtilen aynı kanalda (pin'de) her okuma yapmada yeniden bu fonksiyonu kullanmaya gerek yoktur. Her okuma komutunda denetleyici en son bildirilen kanaldaki analog sinyali dijitalle çevireceğini bilir. Başka bir kanaldaki (pin'deki) analog sinyali dijitalle çevirmek istediğimizde tekrardan bu fonksiyonu kullanarak yeni okunacak kanalı denetleyiciye bildirmemiz gerekmektedir. Bir ADC okuma işlemi bitmeden kanal değiştirme işlemi yapmamalıyız. Bunun için okuma komutundan sonra, ADC çevriminin yapılabilmesi için yaklaşık 10-20 μ s'n'lik bir gecikme vermek veya ADC çevrim işleminin bitip bitmediğini kontrol etmek gereklidir.

• **READ_ADC () FONKSİYONU**

- Bu fonksiyon belirtilen kanaldan alınan sinyalin dijital bilgi karşılığını okur ve o değerle geri döner. Önceden anlatılan "setup_adc(), setup_adc_ports(), set_adc_channel()" fonksiyonlarının önceden tanımlanması gerekmektedir.
- değişken ismi = read_adc();
- setup_adc(ADC_CLOCK_INTERNAL);
- setup_adc_ports(ALL_ANALOG);
- set_adc_channel(1);
- bilgi=read_adc();
- Yukarıdaki örnekte AN1 girişindeki analog sinyal dijitale çevriliyor ve "bilgi" adındaki değişkene aktarılıyor.
- READ_ADC() fonksiyonunda, parantez içine belli "mod" sabitleri de yazılarak kullanılabilir.

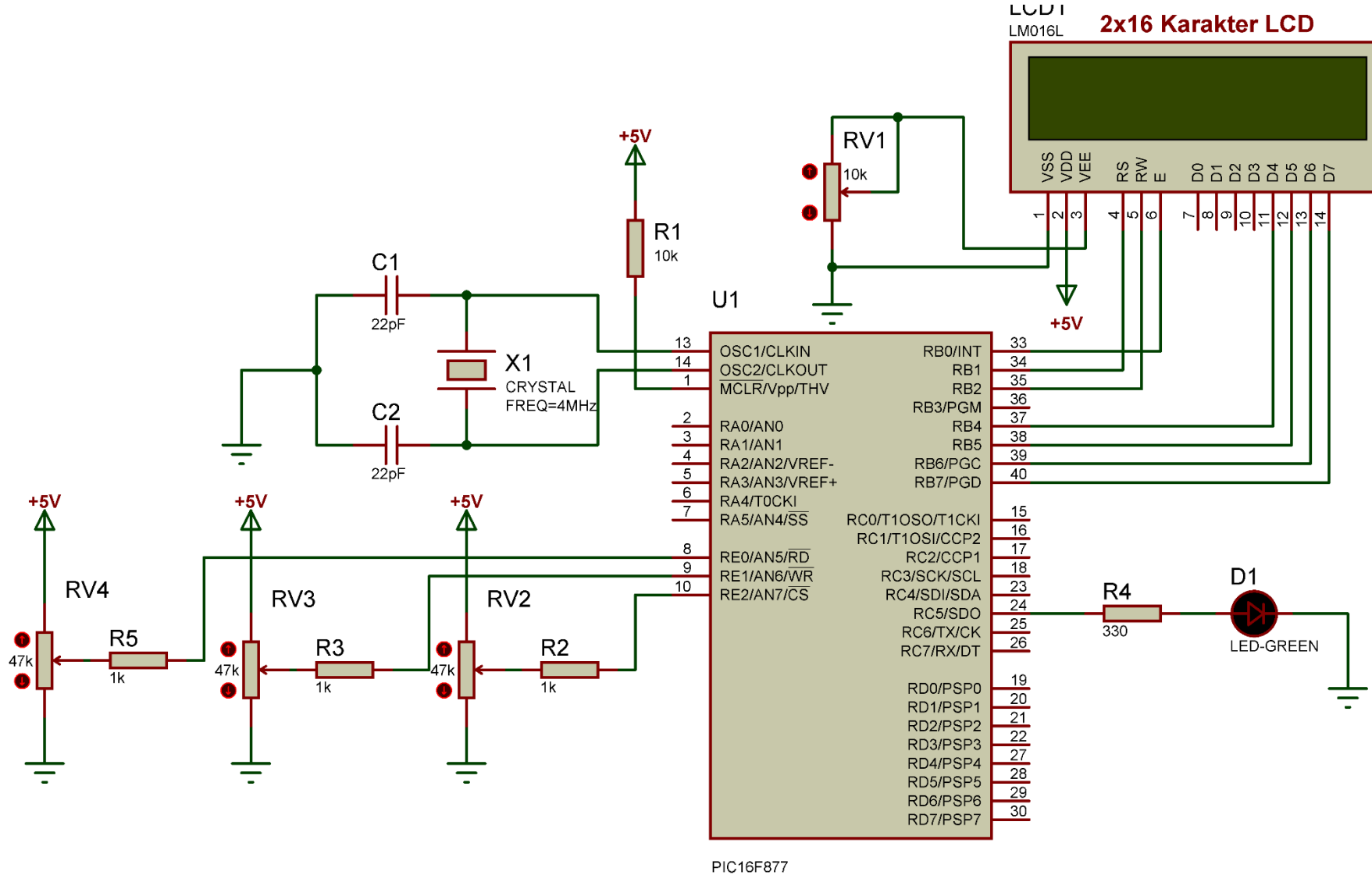
- **READ_ADC () FONKSİYONU**

- `read_adc (ADC_START_AND_READ);` //Normal kullanımdır. Hiçbir yazılmasa bu mod yazılmış gibidir. ADC işlemine başlar ve çevrimi yapar.
- `read_adc (ADC_START_ONLY);` // Sadece AD çevrimi işlemini yapar.
- `read_adc (ADC_READ_ONLY);` // Son çevrim işlemi sonucunu okur ve o değerle geri gelir.
- Örnek: `bilgi=read_adc (ADC_READ_ONLY);` // Son ADC çevrimi sonucunu okur.

- **#DEVICE ADC=X KOMUTU**

- Kullanılan mikrodenetleyicinin özelliğine göre ADC modülünün kaç bit olduğu (8 bit, 10 bit, 11 bit, 16 bit) değişir. Örneğin PIC16F877 denetleyicisinde 10 bit'lik A/D modülü vardır. Fakat biz programımızda A/D biriminin 8 bit'lik olarak işlem görmesinin isteyebiliriz. Bu komut bu amaç için kullanılır. Eğer biz derleyiciye kullanılacak A/D modülünün bit sayısını vermesek, derleyici otomatik olarak bir değer atar. Derleyiciye kullanılacak ADC'nin kaç birlik olacağını bildirmek için programın başına aşağıdaki komutlardan biri yazılmalıdır. Bu komutları programa denetleyici dosyasının tanıtım komutundan (`#include <16f877.h>`) sonra yazılmalıdır.
- **#DEVICE ADC=8, #DEVICE ADC=10, #DEVICE ADC=11, #DEVICE ADC= 16**

Örnek Programlar (ADC Uygulaması_1)



Örnek Programlar (ADC Uygulaması_1)

```
1  #include <16f877.h>
2  #device ADC=10
3  #fuses XT,NOWDT
4  #use delay (clock=4000000)
5  #use fast_io(c)
6  #use fast_io(e)
7  #define use_portb_lcd TRUE
8  #include <lcd.c>
9
10 #INT_AD // ADC çevrimi bitti kesmesi
11 void ADC_Kesmesi ( )
12 {
13     output_high(pin_c5); // RC5 çıkışı 1
14     delay_ms(200);
15     output_low(pin_c5); // RC5 çıkışı 0
16 }
17 unsigned long int bilgi;
18 float voltaj;
19
20 void main ( )
21 {
```

Örnek Programlar (ADC Uygulaması_1)

```
20 void main ( )
21 {
22     setup_psp(PSP_DISABLED);           // PSP birimi devre dışı
23     setup_timer_1(T1_DISABLED);        // T1 zamanlayıcısı devre dışı
24     setup_timer_2(T2_DISABLED,0,1);    // T2 zamanlayıcısı devre dışı
25     setup_CCP1(CCP_OFF);               // CCP1 birimi devre dışı
26     setup_CCP2(CCP_OFF);               // CCP2 birimi devre dışı
27
28     set_tris_c(0x00);                  // C portu komple çıkış
29     set_tris_e(0x0F);                  // E portu komple giriş
30
31     output_c(0x00);                    // C portu çıkışını sıfırla
32
33     setup_adc(adc_clock_div_32);        // ADC clock frekansı fosc/32
34     setup_adc_ports(ALL_ANALOG);        // Tüm AN girişleri analog
35     enable_interrupts(INT_AD);           // AD çevrimi bitti kesmesi tanıtılıyor
36     enable_interrupts(GLOBAL);          // Tüm kesmeler aktif
37
38     lcd_init();                         // LCD hazır hale getiriliyor
39
```

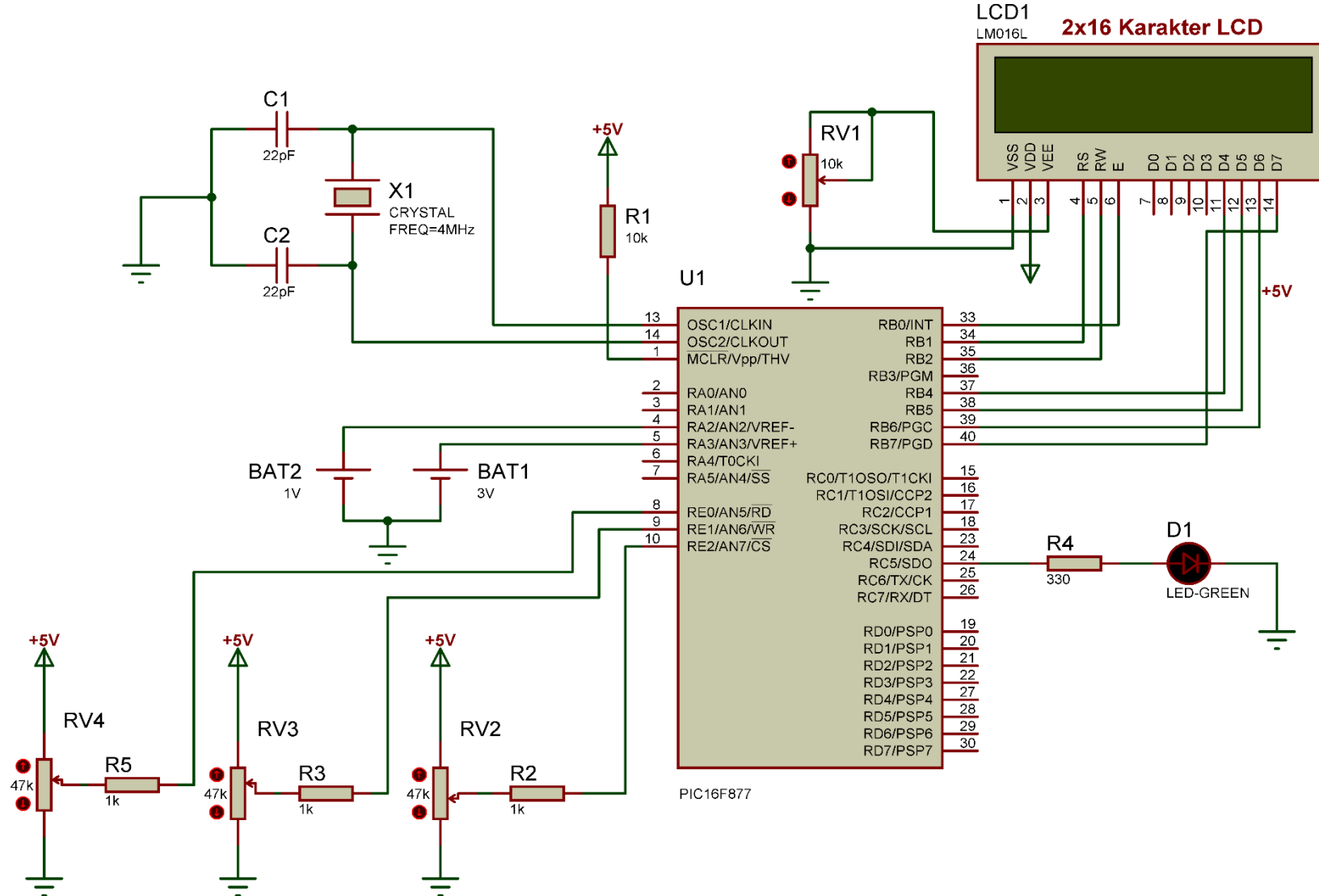
Örnek Programlar (ADC Uygulaması_1)

```
40 printf(lcd_putc, "\f ADC UYGULAMASI "); // LCD'ye yazı yazdırılıyor
41 delay_ms(1500);
42
43 while(1) // sonsuz döngü
44 {
45     set_adc_channel(5); // RE0/AN5 ucundaki sinyal A/D işlemine taşınır
46     delay_us(20); // Kanal seçiminde sonra bu bekleme süresi
47     bilgi=read_adc(); // ADC sonucu okunuyor ve bilgi değişkenine atanıyor
48
49     voltaj=0.0048828125*bilgi; // Dijitale çevirme işlemine uğrayan
50
51     printf(lcd_putc, "\fAN5 Kanali");
52     delay_ms(1500);
53     printf(lcd_putc, "\fDijital=%lu", bilgi); // AN5 ucundaki sinyal
54     printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN5 ucundaki sinyal
55     delay_ms(2500);
56
57     set_adc_channel(6); // RE1/AN6 ucundaki sinyal A/D işlemine taşınır
58     delay_us(20); // Kanal seçiminde sonra bu bekleme süresi
59     bilgi=read_adc(); // ADC sonucu okunuyor ve bilgi değişkenine atanıyor
60
```

Örnek Programlar (ADC Uygulaması_1)

```
60
61     voltaj=0.0048828125*bilgi; // Dijitale çevirme
62
63     printf(lcd_putc, "\fAN6 Kanali");
64     delay_ms(1500);
65     printf(lcd_putc, "\fDijital=%lu", bilgi); // AN6
66     printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN6
67     delay_ms(2500);
68
69     set_adc_channel(7); // RE2/AN7 ucundaki sinyal
70     delay_us(20); // Kanal seçiminde sonra l
71     bilgi=read_adc(); // ADC sonucu okunuyor ve
72
73     voltaj=0.0048828125*bilgi; // Dijitale çevirme
74
75     printf(lcd_putc, "\fAN7 Kanali");
76     delay_ms(1500);
77     printf(lcd_putc, "\fDijital=%lu", bilgi); // AN7
78     printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN7
79     delay_ms(2500);
80 }
81 }
```

Örnek Programlar (ADC Uygulaması 2)



Örnek Programlar (ADC Uygulaması_2)

```
1  #include <16f877.h>
2  #device ADC=10
3  #fuses XT,NOWDT
4  #use delay (clock=4000000)
5  #use fast_io(c)
6  #use fast_io(e)
7  #define use_portb_lcd TRUE
8  #include <lcd.c>
9
10 #INT_AD // ADC çevrimi bitti kesmesi
11 void ADC_Kesmesi ( )
12 { output_high(pin_c5); // RC5 çıkışı 1
13   delay_ms(200);
14   output_low(pin_c5); // RC5 çıkışı 0
15 }
16
17 unsigned long int bilgi;
18 float voltaj;
19
20 void main ( )
```

Örnek Programlar (ADC Uygulaması 2)

```
20 void main ( )
21 { setup_psp(PSP_DISABLED);          // PSP birimi devre dışı
22   setup_timer_1(T1_DISABLED);        // T1 zamanlayıcısı devre dışı
23   setup_timer_2(T2_DISABLED,0,1);    // T2 zamanlayıcısı devre dışı
24   setup_CCP1(CCP_OFF);               // CCP1 birimi devre dışı
25   setup_CCP2(CCP_OFF);               // CCP2 birimi devre dışı
26
27   set_tris_c(0x00);                 // C portu komple çıkış
28   set_tris_e(0x0F);                 // E portu komple giriş
29
30   output_c(0x00);                   // C portu çıkışını sıfırla
31
32   setup_adc(adc_clock_div_32);       // ADC clock frekansı fosc/32
33   setup_adc_ports(AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF); //ADC girişleri ayarlanıyor
34   enable_interrupts(INT_AD);          // AD çevrimi bitti kesmesi tanıtılıyor
35   enable_interrupts(GLOBAL);         // Tüm kesmeler aktif
36
37   lcd_init();                        // LCD hazır hale getiriliyor
38
39   printf(lcd_putc,"\f ADC UYGULAMASI "); // LCD'ye yazı yazdırılıyor
40   delay_ms(1500);
41 }
```


Örnek Programlar (ADC Uygulaması_2)

```
42 while(1)    // sonsuz döngü
43 { set_adc_channel(5);    // RE0/AN5 ucundaki sinyal A/D işlemine tal
44   delay_us(20);          // Kanal seçiminde sonra bu bekleme süresi
45   bilgi=read_adc();      // ADC sonucu okunuyor ve bilgi değişkenine
46
47   voltaj=1+(0.001953125*bilgi);    // Dijitale çevirme işlemine uğra
48
49   printf(lcd_putc, "\fAN5 Kanali");
50   delay_ms(1500);
51   printf(lcd_putc, "\fDijital=%lu", bilgi); // AN5 ucundaki sinyalin
52   printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN5 ucundaki sinyalin
53   delay_ms(2500);
54
55   set_adc_channel(6);    // RE1/AN6 ucundaki sinyal A/D işlemine tal
56   delay_us(20);          // Kanal seçiminde sonra bu bekleme süresi
57   bilgi=read_adc();      // ADC sonucu okunuyor ve bilgi değişkenine
58
59   voltaj=1+(0.001953125*bilgi);    // Dijitale çevirme işlemine uğray
60
```


Örnek Programlar (ADC Uygulaması_2)

```
60
61     printf(lcd_putc, "\fAN6 Kanali");
62     delay_ms(1500);
63     printf(lcd_putc, "\fDigital=%lu", bilgi); // AN6 ucundaki s
64     printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN6 ucundaki s
65     delay_ms(2500);
66
67     set_adc_channel(7); // RE2/AN7 ucundaki sinyal A/D işler
68     delay_us(20); // Kanal seçiminde sonra bu bekleme
69     bilgi=read_adc(); // ADC sonucu okunuyor ve bilgi değ:
70
71     voltaj=1+(0.001953125*bilgi); // Dijitale çevirme işlemi
72
73     printf(lcd_putc, "\fAN7 Kanali");
74     delay_ms(1500);
75     printf(lcd_putc, "\fDigital=%lu", bilgi); // AN7 ucundaki s
76     printf(lcd_putc, "\nVoltaj=%fV", voltaj); // AN7 ucundaki s
77     delay_ms(2500);
78 }
79 }
```

Kaynaklar

- CCS C Programlama Kitabı, Serdar Çiçek, Altaş Yayıncılık
- Mikroelektronika C programlama e-kitabı «<https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c>»
- Mikroişlemciler Lab. EE-304 dersi deney föyleri, Prof. Dr. Mehmet DEMİRTAŞ