BMS 302 MİKROKONTROLCÜLER

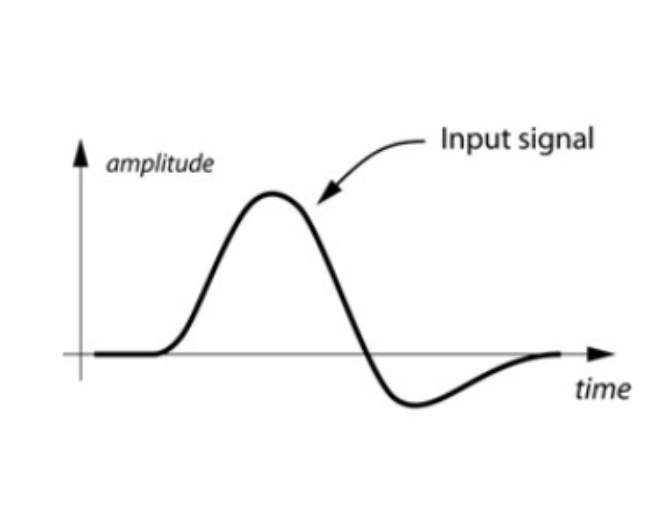
ANALOG/DİJİTAL ÇEVİRME

Prof.Dr. Mutlu AVCI

Çukurova Üniversitesi Biyomedikal Mühendisliği Bölümü

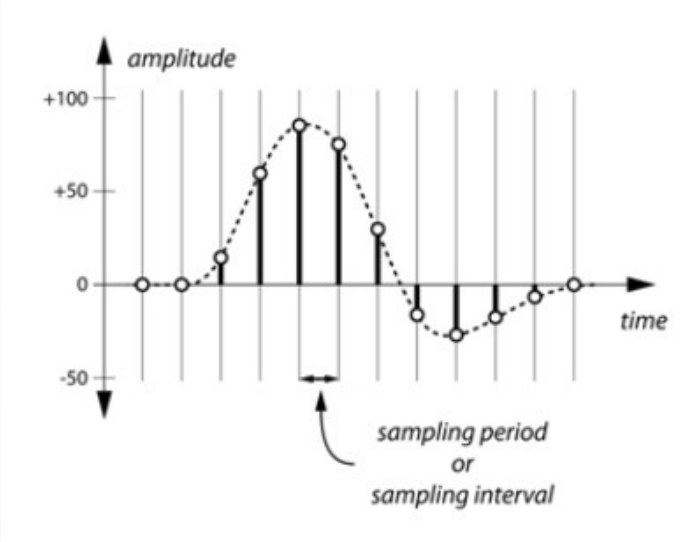
Bahar 2020

- Analog işaret zamana göre devamlıdır. İki nokta arası sonsuz sayıda nokta yada veri vardır.

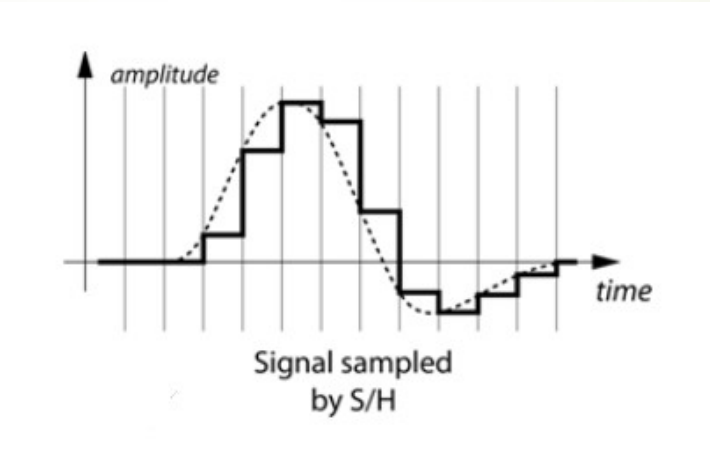


- Analog işareti herhangi bir bellekte tutma şansımız yoktur.

- Veriyi ilk önce örneklemek (sampling) gerekir. Böylece ayrık (discrete) veri elde edilmiş olur. Örnekleme eşit aralıklarla veri almak diğer verileri dikkate almamak işlemidir.

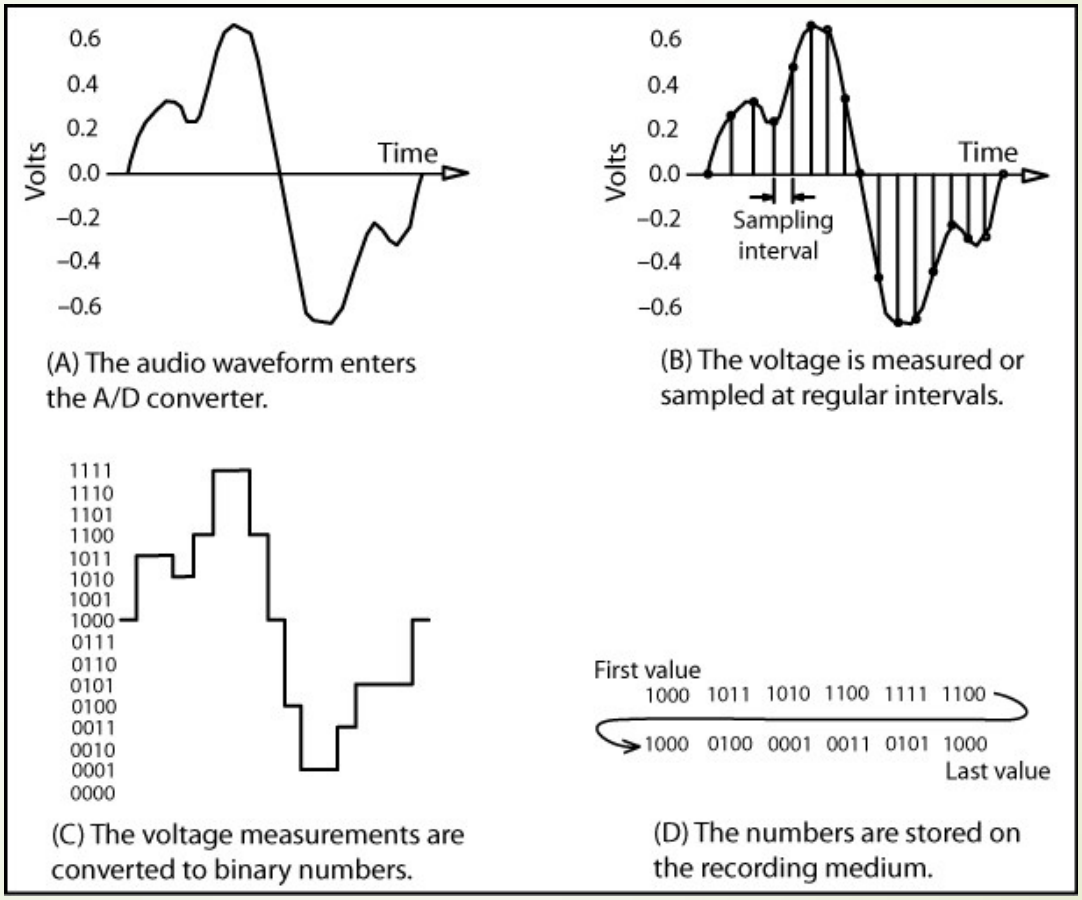


- Örneklenen verilerin aralıkları boş olmamalı bir sonraki veriye kadar sabit olmalıdır. Bu işleme tutma(hold) denir. Örnekleme ve tutma aynı birim tarafından eş zamanlı yapılır, bunu yapan birime S/H (sample/hold) birimi denir. Tutma sonrası veri aşağıdaki hale gelir.



Signal sampled
by S/H

- Son adım kodlama adımıdır. Bu adım verinin sayısal olarak ifade edilmesidir. Bu adımda genellikle ikili sayı sistemi kullanılır. Tüm işlemleri özetlersek:



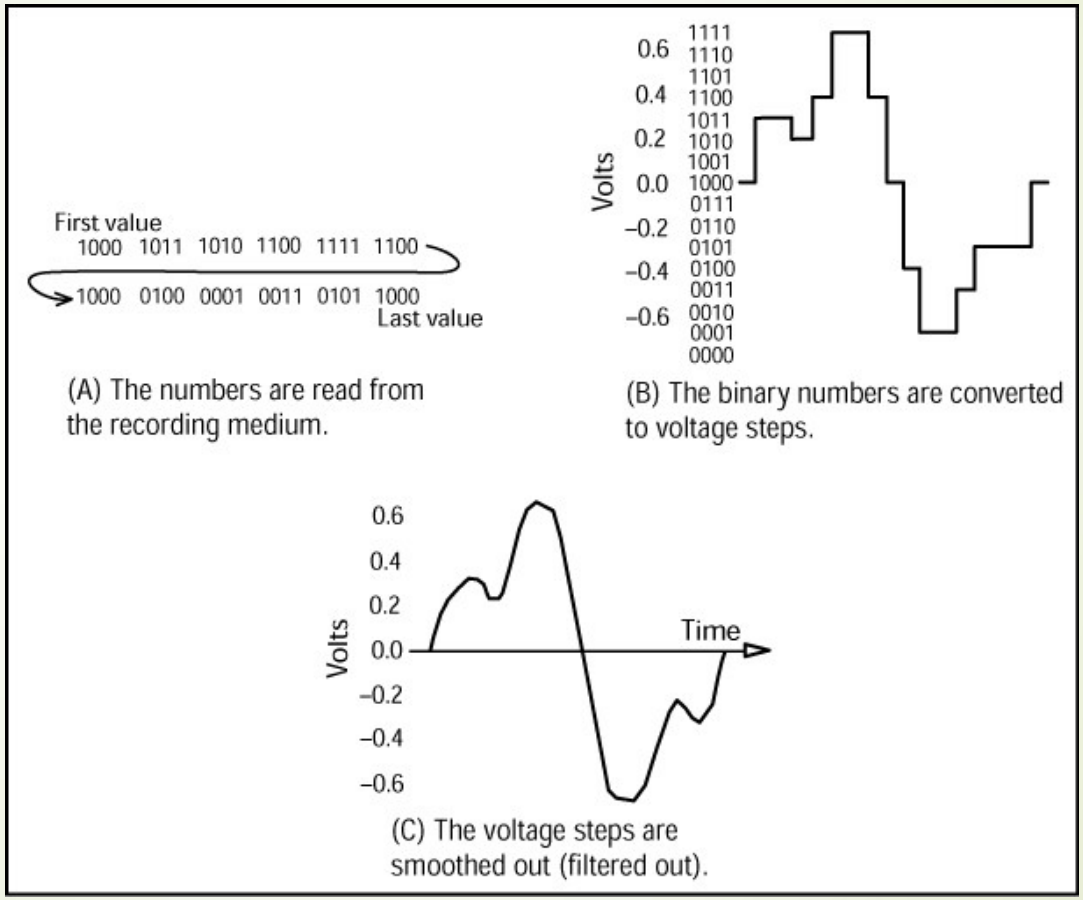
(A) The audio waveform enters the A/D converter.

(B) The voltage is measured or sampled at regular intervals.

(C) The voltage measurements are converted to binary numbers.

(D) The numbers are stored on the recording medium.

- Sayısaldan Analog'a Dönüştürme işlemleri ise aşağıdaki şekil üzerinde özetlenebilir.



- Örnekleme frekansı işaretin tekrar üretilebilmesi için işaret frekansının en az iki katı olmalıdır. Buna Nyquist frekansı denir. Kaşifi olan Harry Nyquist'in adı ile anılmaktadır.

- Mikrokontrolcümüzün içerisinde yer alan Analogtan Sayısala (A/D) dönüştürücü modül sayesinde Analog verileri alıp sayısala dönüştürülmüş hallerini tamamen yazılım kabiliyeti ile işleyebiliriz.

- Mikrokontrolcümüzün 8 kanal Analog girişi olup bunlar aşağıdaki uçları giriş terminali olarak kullanmaktadır.

- ADC (A/D dönüştürücü) kanal numaraları ve PIC16F877A terminal uçları

Kanal No		Uç Adı
0	=	RA0/AN0 ucu (2 numaralı uç)
1	=	RA1/AN1 ucu (3 numaralı uç)
2	=	RA2/AN2 ucu (4 numaralı uç)
3	=	RA3/AN3 ucu (5 numaralı uç)
4	=	RA5/AN4 ucu (7 numaralı uç)
5	=	RE0/AN5 ucu (8 numaralı uç)
6	=	RE1/AN6 ucu (9 numaralı uç)
7	=	RE2/AN7 ucu (10 numaralı uç)

A/D dönüşümünün temel kavramları

- **Çözünürlük (resolution):** Sayısal kodlama için kullanılacak bit sayısıdır. Örneğin 8-bit çözünürlükte 195 sayısı 1100 0101 şeklinde 8 haneli ikili sayı ile kodlanacaktır. 8-bit çözünürlükte işaretsiz 0 ile 255 arası toplam 256 tane farklı kodlama yapılabilir. Yani 2^8 ayrı sayı oluşturulabilir.

- Bit çözünürlüğümüz 10-bit olursa $2^{10} = 1024$ farklı sayı yazabiliriz.

- **Dönüştürme Aralığı:** Analog verinin hangi aralıkta dönüşüme uğrayacağını gösterir. Üst ve alt dönüştürme referans gerilimlerinin farkıdır.
- Üst dönüştürme gerilimi = Vref(+), Alt dönüştürme gerilimi = Vref(-)

- **Adım büyüklüğü:** 1 bit sayısal verinin Analog olarak karşılığını verir. En küçük değerli bitin (Least Significant Bit – LSB) analog değeri de adım büyüklüğüne eşittir. Diğer bir ifade ile ADC'nin dijital bilgiye dönüştürebildiği en küçük gerilim değerine ***adım büyüklüğü*** denir.

$$\text{Adım büyüklüğü} = \frac{V_{ref(+)} - V_{ref(-)}}{2^n}$$

Burada n bit sayısıdır.

Örnek: 0 ile 5V arası dönüşüm yapan 10 bitlik bir ADC'nin adım büyüklüğünü hesaplayınız.

$$\text{Adım büyüklüğü} = \frac{V_{ref(+)} - V_{ref(-)}}{2^n} = \frac{5-0}{2^{10}} = \frac{5}{1024} = 0.0048828125 \text{ V bulunur.}$$

- **Çevrim Süresi (Conversion Time):** Analog bir işaretin sayısala dönüştürülmesi için gereken süreye çevrim süresi denir.

- **Kuantalama Hatası (Quantization error):** Analogtan sayısala dönüşüm sırasında en küçük değerli son bitte (LSB) oluşan hatadır. Maksimum ±½ LSB değerindedir.

- **Analogtan sayısala dönüşüm hesabı:**

Sayısal_değer = (Analog_deger – Vref(-))/Adım_Büyüklüğü

- Benzer şekilde **sayısaldan analoğa dönüşüm hesabı:**

Analog_değer = Sayısal_değer * Adım_Büyüklüğü + Vref(-)

Örnek: Bir PIC16F877A mikrodenetleyici 2 ile 4V arası 10-bit çözünürlükle dönüşüm yapmaktadır. Buna göre;

a) 2.786 V gerilim değerinin sayısal karşılığı nedir? Quantalama hatası?

b) Sayısal 1011100011 değeri analog kaç V gerilime karşılık gelir?

a) Adım_Büyüklüğü = $\frac{4-2}{2^{10}}$ = 0.001953125

Sayısal_değer = (Analog_değer – Vref(-)) / Adım_Büyüklüğü

Sayısal_değer = (2.786 – 2) / 0.001953125

Sayısal_değer = 402.432 = 402

(402.5 ve üstü olsaydı 403'e yuvarlanırdı)

Quantalama Hatası = 0.432 * Adım_Büyüklüğü = 0.432 * 0.001953125

Quantalama Hatası = 0.00084375 V bulunur.

b) Analog_değer = Sayısal_değer * Adım_Büyüklüğü + Vref(-)

Sayısal_değer = 1011100011 = 512 + 128 + 64 + 32 + 2 + 1 = 739

Analog_değer = 739 * 0.001953125 + 2 = 3.443359375 V bulunur.

Örnek : Örnek: Bir PIC16F877A mikrodenetleyici -2 ile 4V arası 10-bit çözünürlükle dönüşüm yapmaktadır. Buna göre;

a) 1.756 V gerilim değerinin sayısal karşılığı nedir? Quantalama hatası?

b) Sayısal 0011100011 değeri analog kaç V gerilime karşılık gelir?

a) Adım_Büyüklüğü = $\frac{4-(-2)}{2^{10}}$ = 0.005859375

Sayısal_değer = (Analog_değer – Vref(-)) / Adım_Büyüklüğü

Sayısal_değer = (1.756 + 2) / 0.005859375

Sayısal_değer = 641.024 = 641

Quantalama Hatası = 0.024 * Adım_Büyüklüğü = 0.024 * 0.005859375

Quantalama Hatası = 0.000140625 V bulunur.

b) Analog_değer = Sayısal_değer * Adım_Büyüklüğü + Vref(-)

Sayısal_değer = 0011100011 = 128 + 64 + 32 + 2 + 1 = 227

Analog_değer = 227* 0.005859375 - 2 = -0.669921875 V bulunur.

PIC16F877A için ADC kullanımı

SETUP_ADC( ) Fonksiyonu

Bu fonksiyon ile A/D biriminin mod ayarı yapılır. ADC'nin kapalı olup olmadığı, ADC frekansı olarak mikro kontrolcünün kendi frekansı mı yoksa harici bir frekans mı kullanılacağı, kendi frekansı kullanılacaksa, kaça bölünerek kullanılacağı gibi işlemler bu fonksiyonla belirtilir.

setup_adc (mod) ;

şeklindedir. Fonksiyonun "mod" kısmına aşağıda belirtilen sabit tanımlamalardan biri yazılabilir.

- setup_adc(ADC_OFF) // ADC biriminin kullanılmayacağını belirtir.
- setup_adc(ADC_CLOCK_INTERNAL) // ADC çalışması için saat sinyali, mikro kontrolcü içerisindeki // osilatörden alınacaktır. fADC = f osc
- setup_adc(ADC_CLOCK_DIV_2) // ADC çalışma frekansı fADC = f osc / 2
- setup_adc(ADC_CLOCK_DIV_8) // ADC çalışma frekansı fADC = f osc / 8
- setup_adc(ADC_CLOCK_DIV_32) // ADC çalışma frekansı fADC = f osc / 32

SETUP_ADC_PORTS ( ) Fonksiyonu

ADC birimine sahip mikro kontrolcülerde sadece A/D çevirimi için kullanılan özel giriş portları yoktur.

Genel giriş çıkış portları hem normal kullanım hem de A/D Çevrimi için kullanılır.

Fakat A/D çevirimi için kontrolcüde hangi uçların kullanılacağı bellidir.

Program yazımı sırasında yapılması gereken bu uçlardan hangisinin A/D çeviriminde analog giriş için, hangisinin normal dijital giriş/çıkış işlemi için kullanılacağının belirtilmesidir.

Bunun için ***setup_adc_ports ( )*** fonksiyonu ile kullanılacak analog giriş uçları derleyiciye bildirilir. Aynı zamanda bu fonksiyon ile A/D çevriminde kullanılacak Vref(+) ve Vref(-) voltaj giriş uçları belirlenir.

setup_adc_ports(sabit tanımlar);

Fonksiyonun "*sabit tanımlar*" kısmına aşağıda belirtilen tanımlari yazılabilir.

- "ALL_ANALOG" = A/D çevrimi için ayrılan tüm uçların analog giriş için kullanılacağını ve VDD beslemesinin Vref(+) VSS beslemesinin ise Vref(-) olarak kullanılacağını belirtir.

- "NO_ANALOGS" = A/D çevirimi için ayrılan hiçbir ucun 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, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için VSS beslemesinin kullanılacağını belirtir.

- "AN0_AN1_AN2_AN3_AN4" = RA0, RA1, RA2, RA3 ve RA5 girişlerinin analog giriş olacağını ve VDD beslemesinin Vref(+) VSS beslemesinin ise Vref(-) olarak kullanılacağını belirtir.

- “AN0_AN1_AN2_AN4_VSS_VREF” = RA0, RA1, RA2 ve RA4 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için VSS beslemesinin kullanılacağını belirtir.

- “AN0_AN1_AN3” = RA0, RA1, RA3 girişlerinin analog giriş olacağını ve VDD beslemesinin Vref(+) VSS beslemesinin ise Vref(-) olarak kullanılacağını belirtir.

- “AN0_AN1_VSS_VREF” = RA0, RA1 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için VSS beslemesinin kullanılacağını belirtir.

- “AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF” = RA0, RA1, RA5, RE0, RE1 ve RE2 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için RA2 ucundan alınan sinyalin kullanılacağını belirtir.

- “AN0_AN1_AN2_AN3_AN4_AN5” = RA0, RA1, RA2, RA3, RA5 ve RE0 girişlerinin analog giriş olacağını, VDD beslemesinin Vref(+) VSS beslemesinin ise Vref(-) olarak kullanılacağını belirtir.

- “AN0_AN1_AN2_AN4_AN5 _VSS_VREF” = RA0, RA1, RA2, RA5, RE0 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için VSS beslemesinin kullanılacağını belirtir.

- “AN0_AN1_AN4_AN5_VREF_VREF” = RA0, RA1, RA5, RE0 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için RA2 ucundan alınan sinyalin kullanılacağını belirtir.

- “AN0_AN1_VREF_VREF” = RA0, RA1 girişlerinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için RA2 ucundan alınan sinyalin kullanılacağını belirtir.

- “AN0” = RA0 girişinin analog giriş olacağını ve VDD beslemesinin Vref(+) VSS beslemesinin ise Vref(-) olarak kullanılacağını belirtir.

- “AN0_VREF_VREF” = RA0 girişinin analog giriş olacağını, Vref(+) için RA3 ucundan alınan sinyalin, Vref(-) için RA2 ucundan alınan sinyalin kullanılacağını belirtir.

Örneğin:

setup_adc_ports(AN0_AN1_VSS_VREF)

Yukarıdaki örnek ile RA0 ve RA1 uçlarının analog giriş, Vref(+) için RA3 ucundan alınan sinyalin ve Vref(-) için VSS geriliminin kullanılacağı belirtilmiştir.

SET_ADC_CHANNEL ( ) Fonksiyonu

Analog / dijital çevrim yapılacak kanal seçimini yapan fonksiyondur.

Analog sinyali dijital sinyale çevir komutu geldiğinde kontrolcü hangi girişteki (kanaldaki) sinyali çevireceğini bu komut sayesinde bilir.

PIC16F877A'da 8 adet ADC kanalı vardır. Bunlar ;

0 = RA0/AN0 ucu, 1 = RA1/AN1ucu, 2 = RA2/AN2 ucu, 3 = RA3/AN3 ucu

4 = RA5/AN4 ucu, 5 = RE0/AN5 ucu, 6 = RE1/AN6 ucu, 7 = RE2/AN7 ucu

şeklindedir.

set_adc_channel(2)

komutu ile denetleyiciye, çevirme komutu geldiğinde AN2 ucundaki analog girişi dijitale çevirme bilgisi verilir.

Belirtilen aynı kanalda (uçta) her okuma yapmadan yeniden bu fonksiyonu kullanmaya gerek yoktur.

Başka bir kanaldaki analog sinyali dijitale çevirmek istediğimizde tekrar bu fonksiyonu o kanal için kullanarak çevirme işleminin yapılacağı kanalı kontrolcüye bildirmek gerekmektedir.

Bir ADC okuma komutundan sonra ADC çevriminin yapılabilmesi için yaklaşık 10 – 20 µs dönüşüm süresi gerekmektedir.

READ_ADC ( ) Fonksiyonu

- Bu fonksiyon belirtilen kanaldan alınan sinyalin dijital bilgi karşılığını okur ve o değerle geri döner.

- Bu fonksiyonun kullanılabilmesi için, setup_adc( ), setup_adc_ports( ) ve set_adc_channel( ) fonksiyonlarının önceden tanımlanması gerekir.

Kullanımı ;

Değişken adı = read_adc( )

şeklindedir

Örneğin:

```
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(2);
sayı = read_adc( ) ;
```

komutlarıyla, AN2 girişindeki analog sinyal dijitale çevrilir ve sayı adındaki değişkene atanır.

Bu fonksiyon parantez içerisine belirli *mod* sabitleri yazılarak da kullanılabilir.

read_adc(ADC_START_AND_READ) // normal kullanımdır. Parantez içine hiçbir şey yazılmamış ise bu mod geçerli olur. ADC işlemine başlanır ve çevirim yapılır.

read_adc(ADC_START_ONLY) // Sadece ADC çevrim işlemi yapılır.

read_adc(ADC_READ_ONLY) // Son çevirme işlemi değeri okunur ve o değer geri gelir.

#DEVICE ADC = X Komutu

- Bu fonksiyon ile derleyiciye kullanılacak ADC'nin kaç bitlik olduğu belirtiler. Kullanımı ;

#DEVICE ADC = 8

#DEVICE ADC = 10

şeklindedir.

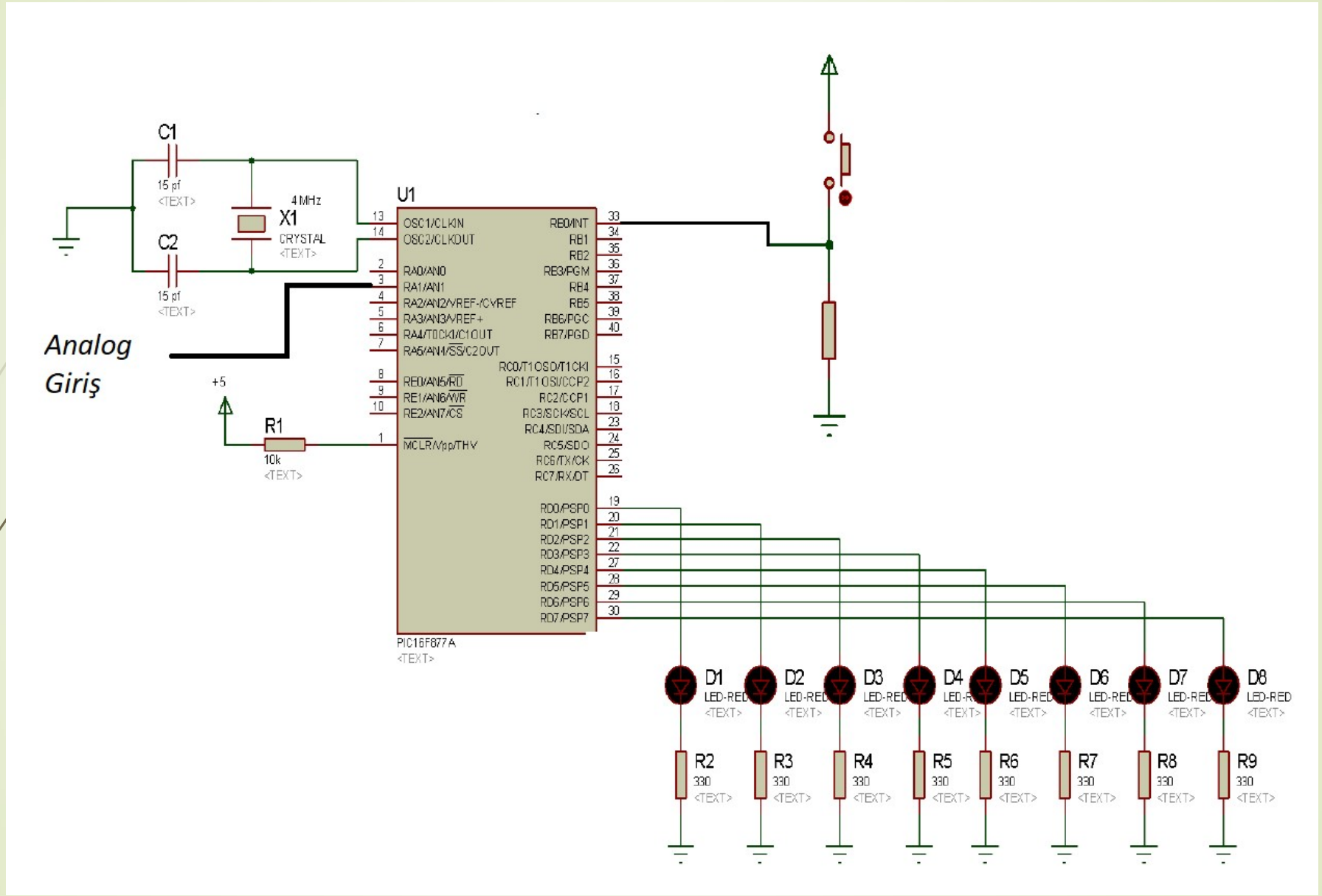
A/D Kesmesi (#INT_AD)

ADC birimi için bir kesme mevcuttur.

A/D işlemi bitince istenirse bir kesme meydana getirilebilir.

#INT_AD kesmesi, A/D birimi analog sinyali dijital bilgiye çevirme işlemini bitirince meydana gelir.

Örnek : PIC16F877A mikrokontrolcüsünün analog girişlerinden RA1'e bağlanmış olan gerilim girişinin sayısal karşılığının ne olduğu D portuna bağlı 8 tane LED lamba ile gösterilmektedir. LED lambanın yandığı uç lojik 1 sönük olduğu uç lojik 0 değerli olacaktır. Bu analogtan sayısala dönüşüm işlemi 0 ile 5V aralığında olup, harici bir push button'a basılınca dönüşüm yapılacak ve çıkış verilecektir. Bir sonraki button'a basışa kadar herhangi bir işlem yapılmayacaktır. Bu işlemi gerçekleştiren ekteki donanım için, CCS C kodunu yazın.


C1
15 pf
<TEXT>
4 MHz
X1
CRYSTAL
<TEXT>
C2
15 pf
<TEXT>
U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A
<TEXT>
Analog
Giriş
+5
R1
10k
<TEXT>
D1
D2
D3
D4
D5
D6
D7
D8
LED-RED
<TEXT>
R2
R3
R4
R5
R6
R7
R8
R9
330
<TEXT>

```
#include <prog.h>
#device ADC=8          // ADC yi 8 bit e ayarladık burası 10 bit te olabiliyor
#use fast_io(A)    // A portundan ADC kullanılacağı için
#use fast_io(B)    // B portundan kesme kullanılacağı için
int8 bilgi;
#INT_EXT
void islem( ){
bilgi=read_adc( );
output_d(bilgi);
}
void main( ){
set_tris_A(0x02);   // AN1 giriş diğerleri çıkış olsun
set_tris_B(0x01);   // RB0 giriş diğerleri çıkış olsun
```

```
setup_adc(adc_clock_div_32);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);

enable_interrupts(INT_EXT);
enable_interrupts(GLOBAL);

while(1);
}
```