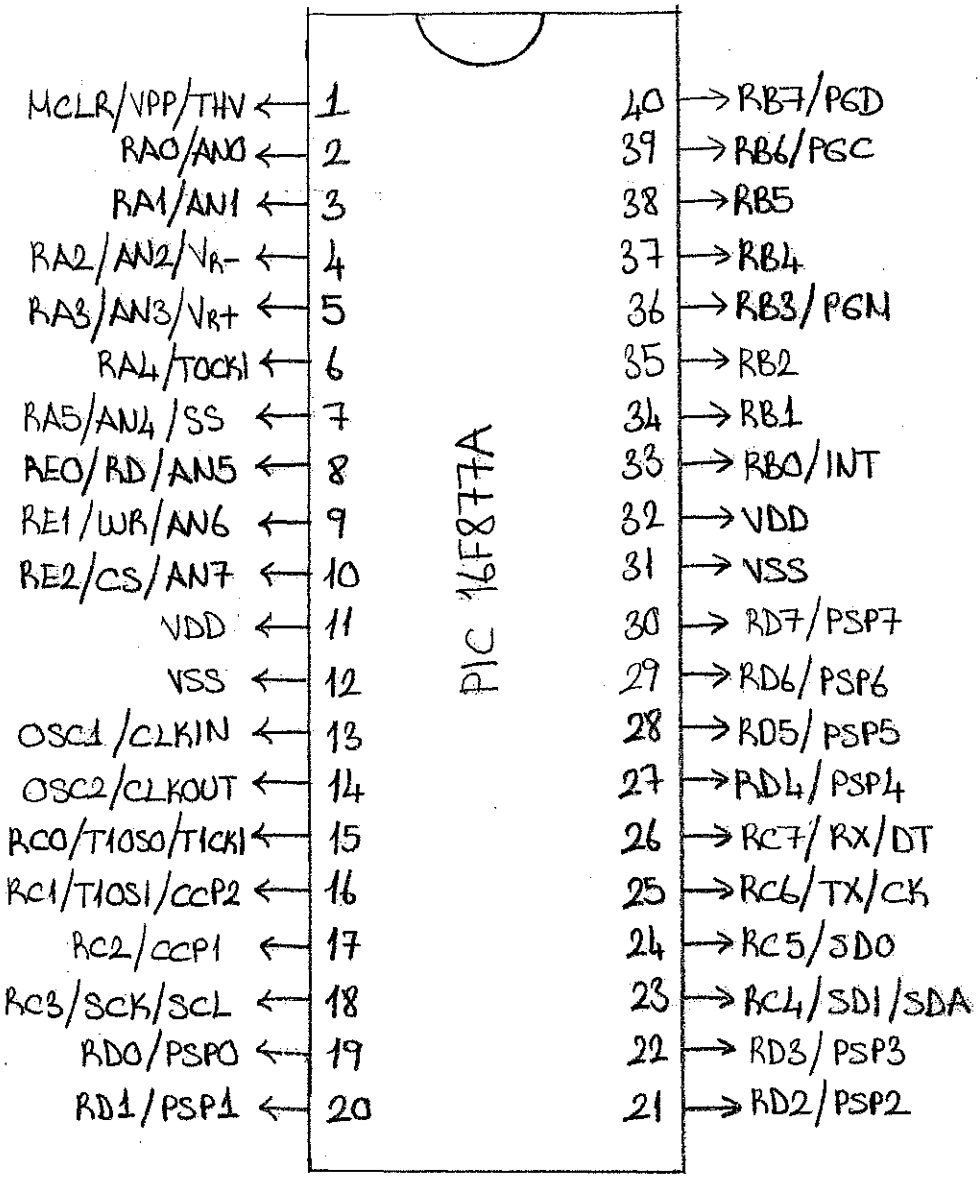
Esra ÇETE

⇐ GİRİŞ ⇒

⇒ PIC 16F877A Mikroişlemcisi



⇒ 16F877A Pin Tanımlamaları

OSC1/CLKIN → 13. Pin → Kristal osilatör girişi / Harici osilatör kaynağı girişi

OSC2/CLKOUT → 14. Pin → Kristal osilatör çıkışı / RC osilatör modunda 1/4f değerinde frekans çıkışı

MCLR/VPP/THV → 1. Pin → Mikrodenetleyici ana reset ucu. Normal çalışmada 1 seviyesinde tutulur.

RA0/AN0 → 2.Pin → A Portu / Analog giriş
RA1/AN1 → 3.Pin → A Portu / Analog giriş
RA2/AN2/V_{REF-} → 4.Pin → A Portu / Analog giriş / Negatif referans gerilimi
RA3/AN3/V_{REF+} → 5.Pin → A Portu / Analog giriş / Pozitif referans gerilimi
RA4/T0CKI → 6.Pin → A Portu / Timer0 için clock girişi
RA5/SS/AN4 → 7.Pin → A Portu / Analog giriş / SSP için slave seçimi

RB0/INT → 33.Pin → B Portu / Harici kesme girişi
RB1 → 34.Pin → B Portu
RB2 → 35.Pin → B Portu
RB3/PGM → 36.Pin → B Portu / Düşük seviye programlama girişi
RB4 → 37.Pin → B Portu
RB5 → 38.Pin → B Portu
RB6/PGC → 39.Pin → B Portu / Seri programlama girişi
RB7/PGD → 40.Pin → B Portu / Seri programlamada data girişi

RC0/T1OSO/T1CKI → 15.Pin → C Portu / Timer1 osilatör çıkışı / Timer1 clock girişi
RC1/T1OSI/CCP2 → 16.Pin → C Portu / Timer1 osilatör girişi / CCP modülü girişi
RC2/CCP1 → 17.Pin → C Portu / CCP modülü girişi
RC3/SCK/SCL → 18.Pin → C Portu / SPI ve I^2C modunda senkron seri clock giriş ve çıkışı
RC4/SDI/SDA → 23.Pin → C Portu / SPI modunda SPI data girişi / I^2C modunda data giriş ve çıkışı
RC5/SDO → 24.Pin → C Portu / SPI modunda SPI data çıkışı
RC6/TX/CK → 25.Pin → C Portu / Usart asenkron gönderme / senkron clock görevi
RC7/RX/DT → 26.Pin → C Portu / Usart asenkron alma / senkron data görevi

RD0/PSP0 → 19.Pin → D Portu / Paralel slave port
RD1/PSP1 → 20.Pin → D Portu / Paralel slave port
RD2/PSP2 → 21.Pin → D Portu / Paralel slave port
RD3/PSP3 → 22.Pin → D Portu / Paralel slave port
RD4/PSP4 → 27.Pin → D Portu / Paralel slave port
RD5/PSP5 → 28.Pin → D Portu / Paralel slave port
RD6/PSP6 → 29.Pin → D Portu / Paralel slave port
RD7/PSP7 → 30.Pin → D Portu / Paralel slave port

RE0/RD/AN5 → 8.Pin → E Portu / PSP okuma kontrolü / Analog giriş
RE1/WR/AN6 → 9.Pin → E Portu / PSP yazma kontrolu / Analog giriş
RE2/CS/AN7 → 10.Pin → E Portu / PSP seçim kontrolü / Analog giriş

VSS → 12, 31. Pinler → Mikrodenetleyici için toprak seviyesi
VDD → 11, 32. Pinler → Mikrodenetleyici için pozitif kaynak gerilimi

⇒ PIC 16F877A Mikrodenetleyicisinin Özellikleri

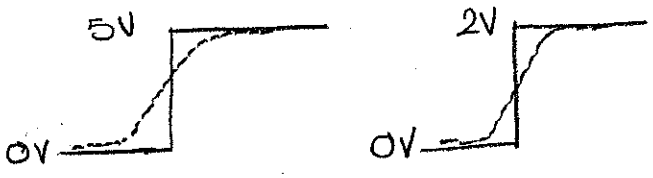
- Yüksek hızlı RISC işlemciye sahiptir.
- 35 tane komutu vardır.
- Tüm komutları 1 cycle, dallanma komutları 2 cycle'da işlenir.
- 20MHz'e kadar işlem hızına sahiptir.
- 8K x 14 word'lük flash program belleği bulunur.
- 368 x 8 bayt'lık data belleği bulunur.
- 256 x 8 bayt'lık EEPROM data belleği bulunur.
- Doğrudan ve dolaylı adresleme yapılabilir.
- Enerji tasarrufu için sleep modu vardır.
- Düşük güçlü, yüksek hızlı CMOSFLASH/EEPROM teknolojisine sahiptir.
- 2V ile 5.5V arasında işlem yapabilir.
- 5V'luk kaynak ile çalışır.

⇒ RISC ve CISC Mimarilerinin Karşılaştırılması

- RISC'de kodlar basittir, CISC'de karmaşıktır.
- RISC'de komutlar sabit 32 bitliktir, CISC'de komutların boyutu sabit değildir.
- CISC'de aynı anda tek bir komut işlenirken, RISC'de ise aynı anda birden fazla komut işlenir. RISC'in bu özelliğine pipeline denir.
- RISC'de program derlenince daha fazla makina kodu olacağından CISC'e göre daha fazla yer kaplar.
- RISC mimarili işlemciler, genellikle aynı saat frekansında çalışan CISC mimarili işlemcilerden daha hızlıdır.

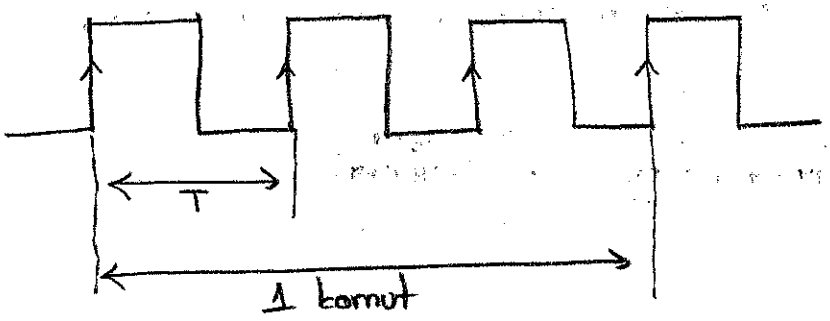
⇒ PIC 16F877A

- Mikroişlemciler düşük voltajda gürültüden daha fazla etkilenirler. PIC, gürültüden en az etkilenen mikroişlemcilerdendir.



0V'dan 2V'a çıkmak, 0V'dan 5V'a çıkmaktan daha kolaydır. Ancak düşük voltajda gürültüden daha fazla etkilenilir.

⇒



Kare dalganın her 4 yükselen kenarında bir komut çalıştırılır.

⇒ Frekansı 4MHz olan bir mikroişlemcide 1 komut ne kadar sürede işlenir?

$F_{osc} = 4MHz \Rightarrow T_{osc} = \frac{1}{F_{osc}} \Rightarrow T_{osc} = \frac{1}{4} = 0,25 \mu sn$

$T_{osc} = 0,25 \mu sn \Rightarrow T_{Komut} = 4 \times T_{osc} \Rightarrow T_{Komut} = 4 \times 0,25 = 1 \mu sn$

⇒ PIC'de ;

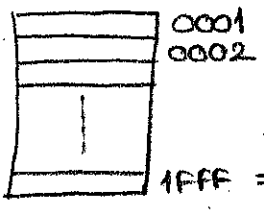
8K×14 word'lük flash memory,
368×8 bayt'lık data memory,
256×8 bayt'lık EEPROM memory bulunur.

Yazılan kodların hex hali flash memory'de tutulur. Değişkenler ve kayıtçılar ise data memory'de tutulur.

RAM enerji kesildiğinde silinir ama EEPROM enerji kesildiğinde silinmez bu nedenle elektrik kesildiğinde silinmemesini istediğimiz verileri EEPROM'a kaydetmeliyiz. (Ör; Şifreli kapı kilidinin şifresini EEPROM'a kaydedebiliriz.)

⇒ PIC'de 8 seviyeli stack vardır. Ya ise en fazla 8 alt programa dallanabiliriz. Ya ise daha fazla alt programa dallanırsak stack'e ilk atılan adres silinir.

⇒ Bir komutun uzunluğu 14 bittir. Flash Memory'nin boyutu 8K×14 olduğundan 14 bitlik makina kodundan 8K kadar tutulabilir.



⇒ 4096 → yani 4096 tane makina kodu tutulabilir.

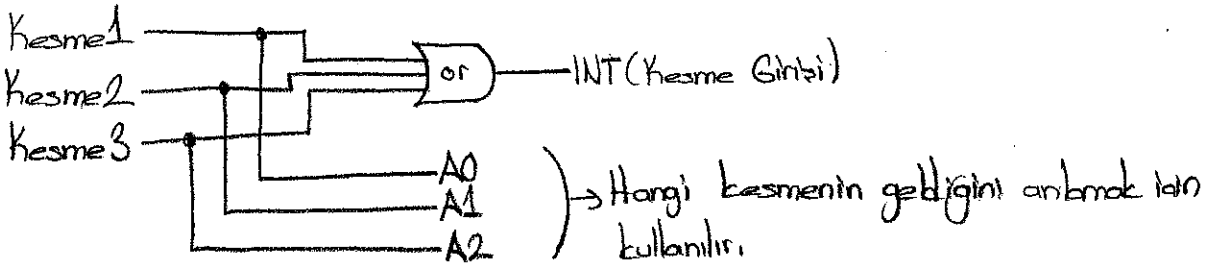
⇒ KESME ⇒ Program çalışırken dışarıdan bir sinyal geldiğinde, kesme alt programına dallanır. Diğer tüm işlerini bırakıp gelen kesmeyle ilgilenir.

⇒ Normal alt programlara dallanma komutuyla gidilir, kesme alt programına kesme düştüğünde gidilir.

⇒ Aynı anda aynı assembler komut cycle'ında 3 sayı toplanamaz. Çünkü ALU'nun iki girişi vardır.

⇒ Program Counter → Bir sonraki çalıştırılacak komutun flash memory'deki adresini tutar, 13 bittir. Çünkü flash memory 8K'dır. Program counter, flash memory'deki adresi tuttuğundan 8K'lık (8×1024) veri 13 bit ile tutulur.

⇒ Aynı anda 3 kesmeye ihtiyaç duyarsak;

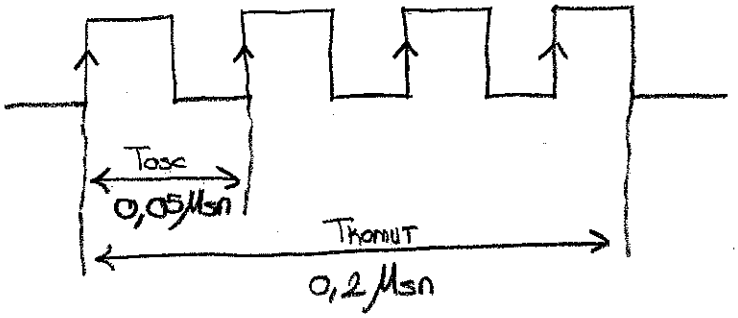


⇒ 1K'lık hafıza ⇒ $1\times1024 = 2^{10}$ ⇒ 10 bit ile tutulur.
2K'lık hafıza ⇒ $2\times1024 = 2^{11}$ ⇒ 11 bit ile tutulur.
256 byte'lık hafıza ⇒ $256 = 2^8$ ⇒ 8 bit ile tutulur.

⇒ 20MHz'lik kristal için;

1 cycle'da geçen süre ⇒ $\frac{1}{20MHz} = 0,05\,\mu sn$

$T_{komut} = 4T = 4\times0,05\,\mu sn = 0,2\,\mu sn = 200\,nsn.$



⇒ Mikroişlemciye enerji verildiği anda reset vektöründen başlar. Reset vektöründe programın başlangıç adresi tutulur.

⇒ Kesme geldiği anda işlemci interrupt vektöre gider. İnterrupt vektöründe kesme alt programının adresi tutulur.

⇒ Flash memory 4 sayfadan oluşur;

Page0 ⇒ 2K
Page1 ⇒ 2K
Page2 ⇒ 2K
Page3 ⇒ 2K

⇒ Data memory 4 bankdan oluşur. Bank değiştirmek için Status registerının RP0, RP1 bitleri kullanılır.

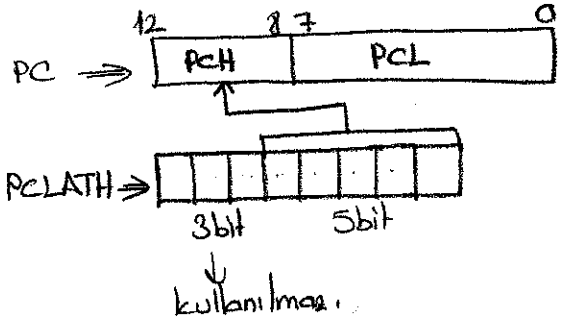
	RP1	RP0
Bank0 ⇒	0	0
Bank1 ⇒	0	1
Bank2 ⇒	1	0
Bank3 ⇒	1	1

⇒ PortA, PortB, PortC, PortD, PortE → Portlarda değer tanımlama işlemleri için,
TrisA, TrisB, TrisC, TrisD, TrisE → Portları, giriş-çıkış yapmak için kullanılır.

⇒ Data memory 4 bankdan oluşur, her bank 128 x8 byte'dır.

128×4 bank = 512 tane kaydedici bulunur. Ancak bunların sadece 368 tanesi aktif olarak genel amaçlı kaydedici olarak kullanılabilir. Geriye kalan 144 kaydedici alanı mikroişlemci tarafından algılanmadığı için kullanılmaz.

⇒ PCL ve PCLATH



PCL, 8 bit olduğundan en fazla 256 kadar ileriye gidebiliriz. Daha uzak adresler için PCH'ı değiştirmek gerekir.
PCL, FF olursa PCH 1 artar. Normalde sadece PCL arttırılır.

⇒ GOTO komutu, sadece PC'in adresini değiştirerek işlem yapar. Bulunduğumuz adresten, dallanacağımız adrese kadar olan mesafeyi hesaplar bu mesafe değerini program counter'a ekler.

Bulunduğumuz adres → 100
Dallanacağımız adres → 156
] Program counter'a 55 eklenir.

⇒ ORG 0x500 ⇒ Bu komutun altındaki kodlar flash'ın 500. adresinden itibaren yazılır.

⇒ kesme

Call Alt_Program → Alt_Program

Retfie

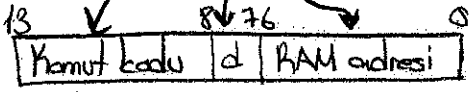
Return

Kesme programı kullanılan başka bir alt programa dallandığımızda, bu alt programdan geri dönerken GOTO KESME komutu kullanılmaz. Return ile tekrar kesme programına gidilir.

⇒ Data memory'de değişkenler enerji kesildiğinde silinir. Enerji tekrar verildiğinde program baştan başlatılır ve değişkenler tekrar yüklenir. Genel amaçlı kaydedilen değişken tanımlamada kullanılır.

⇒ Flash memory'deki makina kodları 14 bittir.

addwf toplam, f



⇒ include P1C16f877A, satırı ile mikrodenetleyiciye özel olan w, f, c gibi önemli bitlerin isimleri otomatik olarak atanır. Böylece bellek adreslerini kullanmak yerine değişkenden isimlerini kullanabiliriz.

⇐ Komutlar ⇒

1⇒ **ADDLW**

ADDLW k → k sabitini working registirinin içeriğine ekler. Sonucu working'e yazar

Etkilenen Bitler → C, DC, Z

2⇒ **ADDWF**

ADDWF f,d → f ile working registerinin içeriğini toplar. Sonucu d=1 ise f registirina, d=0 ise working'e yazar.

Etkilenen Bitler → C, DC, Z

3⇒ **ANDLW**

ANDLW k → k sabiti ile working registerinin içeriğini and'ler. Sonucu working'e yazar.

Etkilenen Bitler → Z

4⇒ **ANDWF**

ANDWF f,d → f ile working registerinin içeriğini and'ler. Sonucu d=1 ise f registerina, d=0 ise working'e yazar.

Etkilenen Bitler → Z

5⇒ **BCF**

BCF f,d → f registerinin b. bitini sıfır yapar.

6⇒ **BSF**

BSF f,d → f registerinin b. bitini bir yapar.

7⇒ **BTFSC**

BTFSC f, b → f registerinin b. bitini test eder. Bu bit 0 ise program bir sonraki komuta atlar.

8⇒ **BTFSS**

BTFSS f,b → f registerinin b. bitini test eder. Bu bit 1 ise program bir sonraki komuta atlar.

9⇒ **CLRF**

CLRF f → f registerinin içeriğini siler.

Etkilenen Bitler → Z

10 ⇒ CLRW

CLRW → Working registerının içeriğini siler.
Etkilenen Bitler → Z

11 ⇒ CALL

CALL k → Program k etiketli alt programa dallanır. PC+1 stack'e kaydedilir.

12 ⇒ GOTO

GOTO k → Program k etiketli alt programa dallanır.

13 ⇒ RETURN

RETURN → Alt programdan ana programa döndürür.

14 ⇒ RETLW

RETLW k → Program interrupt alt programından ana programa döner. k sabitini working'e yükler.

15 ⇒ RETFIE

RETFIE → Program interrupt alt programından ana programa döner.

16 ⇒ DECF

DECF f,d → f registerının içeriğini 1 azaltır. Sonucu d=1 ise f registerına, d=0 ise working'e kaydeder.
Etkilenen Bitler → Z

17 ⇒ INCF

INCF f,d ⇒ f registerının içeriğini 1 arttırır. Sonucu d=1 ise f registirına, d=0 ise working'e yazar.
Etkilenen Bitler ⇒ Z

18 ⇒ DECFSZ

DECFSZ f, d → f registerının içeriğini 1 azaltır. Sonuc 0 ise program bir sonraki komuta atlar.

19 ⇒ INCFSZ

INCFSZ f,d → f registerının içeriğini 1 arttırır. Sonuc 0 ise program bir sonraki komuta atlar.

20 ⇒ MOVLW

MOVLW k → k sabitini workinge yükler.

21 ⇒ MOVWF

MOVWF f ⇒ Workingin içeriğini f registirina yükler.

22 ⇒ IORLW

IORLW k → k sabiti ile workingin içeriğini OR'lar.
Etkilenen Bitler → Z

23 ⇒ IORWF

IORWF f, d → f registeri ile workingin içeriğini OR'lar.
Etkilenen Bitler → Z

24 ⇒ RRF

RRF f, d → f registerinin içeriğini 1 bit sağa kaydırır.
Etkilenen Bitler → C

25 ⇒ RLF

RLF f, d → f registerinin içeriğini 1 bit sola kaydırır.
Etkilenen Bitler → C

26 ⇒ SUBLW

SUBLW k → k sabitinden workingin içeriğini çıkarır.
Etkilenen Bitler → C, DC, Z

27 ⇒ SUBWF

SUBWF f, d → f registirinin içeriğinden workingi çıkarır.
Etkilenen Bitler → C, DC, Z

28 ⇒ XORLW

XORLW k → k sabiti ile workingin içeriğini XOR'lar.
Etkilenen Bitler → Z

29 ⇒ XORWF

XORWF f, d → f registeri ile workingin içeriğini XOR'lar.
Etkilenen Bitler → Z

30 ⇒ SLEEP

SLEEP → İşlemciyi uyku moduna alır.
Etkilenen Bitler → $\overline{TO}$, $\overline{PD}$

31 ⇒ NOP

NOP → Bir komut çevriminde hiçbir işlem yapmaz.

32 ⇒ MOVF

MOVF f,d → f registerının içeriğini d=1 ise f registerına, d=0 ise working'e yollar
Etkilenen Bitler → Z

33 ⇒ CLRWDT

CLRWDT → Watchdog Timer'ı sıfırlar.
Etkilenen Bitler → $\overline{TO}$, $\overline{PD}$

34 ⇒ COMF

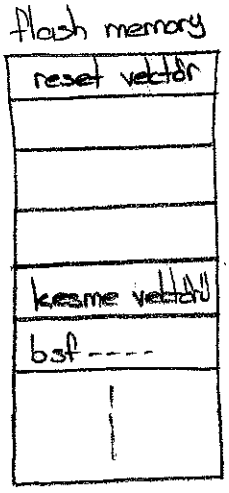
COMF f,d → f registerının içeriğinin tersini alır.
Etkilenen Bitler → Z

35 ⇒ SWAPF

SWAPF f,d → f registerının düşük değerlikli 4 biti ile yüksek değerlikli 4 bitini yer değiştirir.

NOT ⇒ Working, işlemlerde ara al register olarak kullanılır. Örneğin, iki sayının toplanmasında sonuç önce working'e atılır. Çünkü ALU'nun çıkışı working'e gider.

```
⇒ ORG 0X00
  GOTO START
  ORG 0X04
  GOTO KESME
  ORG 0X05
START
  BCF STATUS,5  ⟶
```



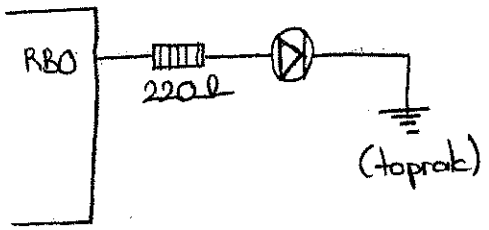
START, 05'den başlayarak hafızaya yazılır. Eğer 00'dan itibaren yazılsaydı reset ve kesme vektörüyle program komutları çakışırdı. Bu nedenle hata oluşurdu.

⇒ Kesme alt programından RETFIE ile geri dönerken, geri dönüş adresi ile flagların eski durumlarında alınır. Böylece kesme gelmeden önceki durumuna geri döner. RETURN'da sadece geri dönüş adresi alınır.

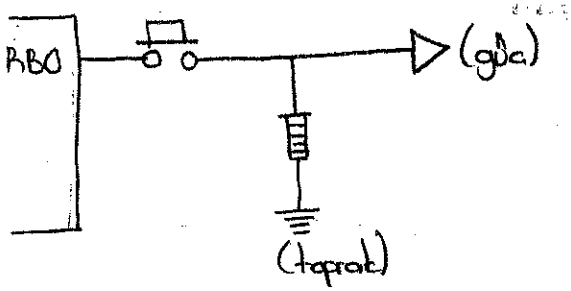
⇒ [redacted]MPLAB'da Proje Oluşturma

- New File'dan yeni dosya oluşturulur. Bu dosya kendi oluşturduğumuz klasör içine ".asm" uzantılı olarak kaydedilir. (deneme.asm)
- Project Wizard'dan yeni proje oluşturulur. Oluşturulan bu projede kendi oluşturduğumuz klasör içine kaydedilir.
- Proje oluşturma sırasında oluşturduğumuz deneme.asm dosyası projeye eklenir.
- Dosya (deneme.asm) projede source files kısmında olmalıdır.
- Deneme.asm dosyasına kodlar yazıldıktan sonra "Build All" butonu ile program çalıştırılır.

Not ⇒ Proteusta Led Bağlama



⇒ Buton Bağlama



⇒ Gecikme Döngüsü

Örnek 1 = 8MHz'lik bir kristali bulunan işlemcide 1ms gecikme oluşturan gecikme alt programı = ?

$f = 8MHz \Rightarrow T = \frac{1}{8} = 0,125 \mu sn$

$T_{KOMUT} = 4T = 4 \times 0,125 = 0,5 \mu sn$

$1ms = 1000 \mu sn$

Gecikme
Movlw D'14' → 1 defa calisir → 0,5 Msn
Movwf Sayac1 → 1 defa calisir → 0,5 Msn
Gecikme1
Movlw D'50' → 14 defa calisir → 14x0,5 Msn
Movwf Sayac2 → 14 defa → 14x0,5 Msn
Gecikme2
Decfsz Sayac2,f → 14x50 defa → 14x50x0,5 Msn
Goto Gecikme2 → 14x50 defa → 14x50x0,5x2 Msn
Decfsz Sayac1,f → 14 defa → 14x0,5 Msn
Goto Gecikme1 → 14 defa → 14x0,5x2 Msn
Return → 1 defa → 1 Msn
+
1087 Msn ≅ 0,1 msn

Örnek2 ⇒ 4MHz'lik bir kristali bulunan işlemcide 30 Msn'lik gecikme=?

Fosc = 4MHz ⇒ Tkomut = 1 Msn

Gecikme
Movlw D'10' → 1 defa calisir → 1 Msn
Movwf Sayac → 1 defa " → 1 Msn
Gecikme1
Decfsz Sayac,f → 10 defa → 10x1 Msn
Goto Gecikme1 → 9 defa → 9x2x1 Msn
Return
30 Msn

⇒ Sayı karşılaştırma İşlemi

- Eşit mi?

→ Mouf Sayı,w
Sublw D'10'
Btfss Status,2
Goto Esit_degil
Goto Esit

→ Mouf Sayı,w
Xorlw D'10'
Btfss Status,2
Goto Esit_degil
Goto Esit

- Büyük - Küçük

```
Movlw D'120'
Subwf Sayı, w
Btfss Status, C
Goto Sayı_120_den_kucuk
Goto Sayı_120_den buyuk
```

Sayı - 120 = negatif ise C biti 0 olur.

⇒ Çıkarma işleminde sonuç negatif ise Carry biti 0 olur.

⇒ Sayı1 - Sayı2 işleminde;

Sayı1 > Sayı2 → Z=0 , C=1
Sayı1 = Sayı2 → Z=1, C=1
Sayı1 < Sayı2 → Z=0, C=0

⇒ #include <p16f877a.inc> → Bu satır yazıldığında adres bloğuna portA, portB, Status, Adcon0 ---- gibi registerların adreslerini yazmaya gerek kalmaz. Registerları doğrudan adlarıyla kullanabiliriz.

Örnek= for i=6; i≥0; i-- döngüsünü kurarak i=0 dunca işlem alt programına giden asm kodu.

```
Movlw D'6'
Movwf i
Kontrol
Decfsz i, f
Goto Kontrol
Goto Islem
```

⇐ Adresleme Metodları ⇒

1⇒ İvedi Adresleme

İvedi adresleme ile working kaine doğrudan değer atanabilir veya içeriğinde değişiklik yapılabilir.

```
movlw b'00001001'
sublw b'00010111'
```

Burada working'e ivedi adresleme ile b'00001001' değeri yüklendi, daha sonra b'00010111' sayısı working'deki değerden çıkarılarak, working'deki değerde değişiklik yapıldı.

İvedi adresleme ile ara değişkenler yerine Working kullanılarak çok daha kısa ve bellekte gereksiz yer işgal etmeyen programlar yazılabilir.

2⇒ Doğrudan Adresleme

Bu metodda komut kaydedicisinin kaine yapılmak istenen işlem ve işlemin uygulanacağı RAM adresi yüklenir. İvedi adreslemede RAM kullanılmazken doğrudan adreslemede kullanılır.

```
movf portB,W
movwf ilkdeger
```

Burada program sayacı "movf portB,W" satırını işaret ettiğinde 14 bitlik bu veri komut kaydedicisi içerisine yüklenir. Komut çözümleyici bir RAM adresini (PortB) doğrudan adresleyerek içeriğinin Working kaydedicisi kaine atılacağını çözümler. 7 bit'lik doğrudan adres veri yolu üzerinden içeriği alınmak istenen kaydedici adresi RAM'e gönderilir. PortB içeriğindeki veri mux üzerinden ALU'ya geçer. ALU'da bir işlem görmeden working'e yüklenir.

Program sayıcı "movwf ilkdeger" satırını gösterdiğinde, komut kaydedicisine yüklenen komut, komut çözümleyici tarafından çözümlenir. RAM, adres veri yolu üzerinden "ilkdeger" isimli 0x21 kaydedicisini işaret eder. Working kaydedicisi içeriği veri yolu üzerinden RAM'e aktarılır. O anda aktif olan RAM adresi 0x21'e veri yazılır.

3⇒ Dolaylı Adresleme=

- Dolaylı adreslemede iki tane kaydedici kullanılır.
- INDF → RAM bellekte adresi bulunan fiziksel bir kaydedici değildir. Bu kaydedici ile FSR'nin gösterdiği adresteki veriye ulaşılır.
- FSR → RAM'de 0X04 adresinde bulunur. FSR'ye daima bir adres verisi yüklenir.

⇒ dizi EQU 0X20 → programın başında daha sonra dolaylı adreslemede kullanılmak üzere dizinin başlangıç adresi tanımlanır.

```
        !
        !
movlw dizi+3    → dizinin 4. elemanının adresi workinge yüklendi.
movwf FSR       → bu adres FSR'ye yazıldı.
movlw 'a'       → workinge 'a' verisi yazıldı.
movwf INDF      → Workingdeki 'a' verisi INDF'ye atıldı.
```

⇒ Aşağıdaki program 0X020 ile 0X02f RAM adresleri arasındaki kaydedicilerin içeriğini siler.

```
        !
        !
movlw 0X20
movwf FSR
Dongu
clrf INDF
incf FSR,f
movlw 0X2f
subwf FSR,W
btfss status,Z
goto Dongu
```

⇒ 20H ile 4DH adresleri arasında kaç tane sıfır olduğunu dolaylı adresleme tekniği ile bulan uygulama

```
sifir equ 0X43
org 0X00
goto basla
org 0X10
```

```
basla
    clrf sifir
    movlw 0X20
    movwf FSR
dongu
    movf INDF,w
    sublw D'0'
    btfsc status,z
    incf sifir;f
    incf FSR,f
    btf status,z
    movlw 0X80
    subwf FSR,W
    btfss status,z
    goto dongu
end
```

⇒ Dolaylı adresleme ile 0X20 ile 0X27 RAM adresleri arasındaki değerleri toplayan uygulama.

```
    movlw 0X20
    movwf FSR
    clrf toplam
dongu
    movf INDF,w
    addwf toplam,f
    incf FSR,f
    movf FSR,w
    sublw 0X27
    btfss status,z
    goto dongu
end
```

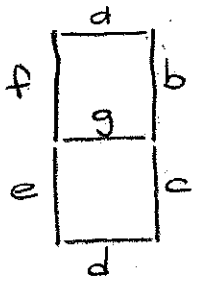
⇒ 10 tane değerin kaydedilmiş olduğu registerların küçükten büyüğe doğru sıralanması uygulaması.

```
basla
    movlw 0X0A      →ilk elemanın adresi
    movwf FSR
dongu
    movlw 0x14      → son elemandan bir sonraki adrese gelindi mi?
    subwf FSR,W
    btfsc status,2
    goto cikis
    incf FSR        →iki elemanı karsılastır.
    movf INDF,W
    incf FSR
    subwf INDF,W
    btfss status,C  →sıralama doğrumu?
    goto dongu      →sıralama doğruysa sıradakine bak.
    movf INDF,W     →sıralama doğru değilse yer değiştir.
    movwf TEMP
    decf FSR
    movf INDF,W
    incf FSR
    movwf INDF
    decf FSR
    movf TEMP,W
    movwf INDF
    goto dongu
cikis
end
```

[redacted]

⇐ DİSPLAY ⇒

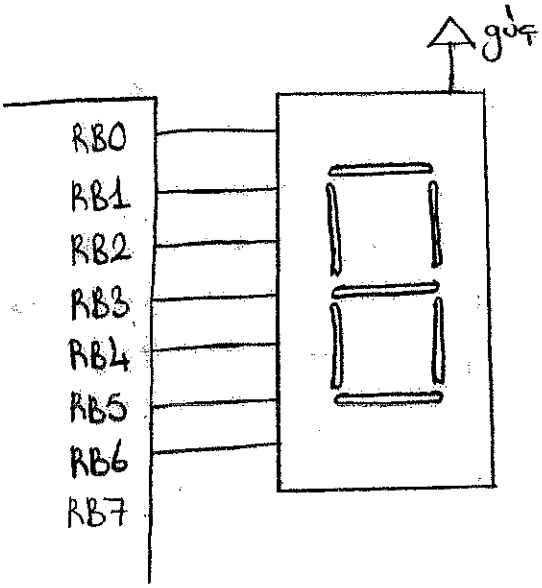
⇒



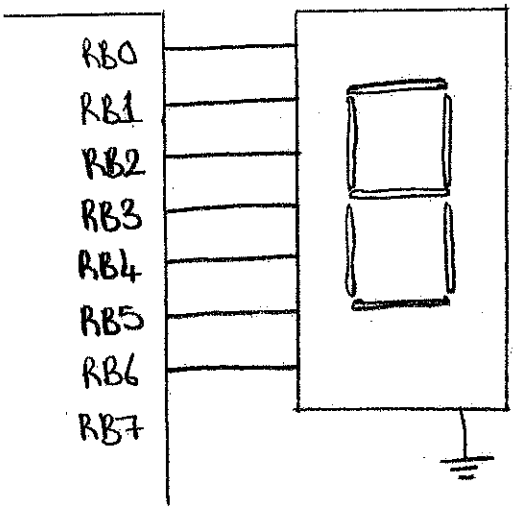
⇒ a, b, c, d, e, f, g displayin segmentleridir.

⇒ Display'ler ortak anotlu ve ortak katotlu olmak üzere iki çeşittir.

- Ortak anotlu display güce, ortak katotlu display toprağa bağlanır.



⇒ Ortak Anotlu Display



⇒ Ortak Katotlu Display

⇒ Ortak anotlu displayde yanacak olan segmente lojik 0, yanmayacak olan segmente ise lojik 1 gönderilir.

⇒ Ortak katotlu displayde yanacak segmente lojik 1, yanmayacak segmente ise lojik 0 gönderilir.

Ortak Anot

	x g f e d c b a	
0→	0 1 0 0 0 0 0 0	⇒0x40
1→	0 1 1 1 1 0 0 1	⇒0x79
2→	0 0 1 0 0 1 0 0	⇒0x24
3→	0 0 1 1 0 0 0 0	⇒0x30
4→	0 0 0 1 1 0 0 1	⇒0x19
5→	0 0 0 1 0 0 1 0	⇒0x12
6→	0 0 0 0 0 0 1 0	⇒0x02
7→	0 1 1 1 1 0 0 0	⇒0x78
8→	0 0 0 0 0 0 0 0	⇒0x00
9→	0 0 0 1 0 0 0 0	⇒0x10

Ortak Katot

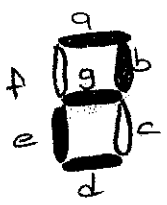
x g f e d c b a	
0 0 1 1 1 1 1 1	⇒0x3f
0 0 0 0 0 1 1 0	⇒0x06
0 1 0 1 1 0 1 1	⇒0x5B
0 1 0 0 1 1 1 1	⇒0x4F
0 1 1 0 0 1 1 0	⇒0x66
0 1 1 0 1 1 0 1	⇒0x6D
0 1 1 1 1 1 0 1	⇒0x7D
0 0 0 0 0 1 1 1	⇒0x07
0 1 1 1 1 1 1 1	⇒0x7F
0 1 1 0 1 1 1 1	⇒0x6F

⇒

x	g	f	e	d	c	b	a
↓	↓	↓	↓	↓	↓	↓	↓
7.bit	6.bit	5.bit	4.bit	3.bit	2.bit	1.bit	0.bit

↳ kullanılmaz.

⇒ 2 Sayısını displayde göstermek için;



⇒

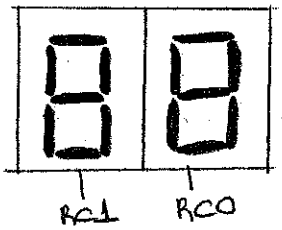
Anot Display

x g f e d c b a
0 0 1 0 0 1 0 0

Katot Display

x g f e d c b a
0 1 0 1 1 0 1 1

⇒ Birden fazla haneli displaylerde display seçme uçları ile display seçilir.



⇒ Bu display anot ise RC0 'a bağlı displayi aktif yapmak için RC0 pinine lojik 1, RC1 pinine lojik 0 göndermeliyiz.

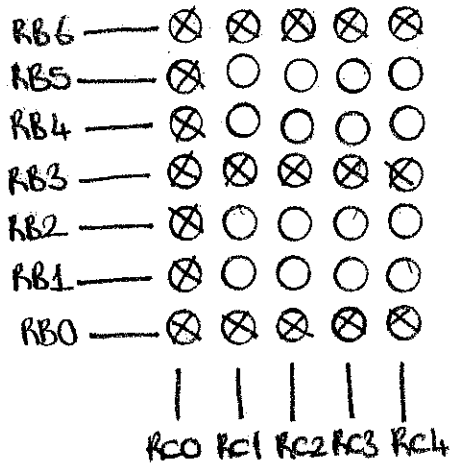
- Ortak anotlu displaylerde aktif edilecek olan displaye lojik 1 gönderilir.
- Ortak katotlu displaylerde aktif edilecek olan displaye lojik 0 gönderilir.

⇒ Birden fazla haneli displaylerde iki displaye aynı anda veri gönderilmez. Önce 1. displaye gönderilir ardından kısacık bir gecikme yapılır daha sonra 2. displaye veri gönderilir. (Silkürme dayı)

⇐ DOT MATRİS ⇒

⇒ Dot matrise sadece rakam değil, istenilen karakter yazılabilir.

⇒ Dot matrise 'E' harfi yazdırmak için;



⇒ Sütün seçme işleminde RC0, RC1, RC2, RC3, RC4 pinlerinden hangisine lojik 1 verilirse o sütuna veri gönderilir.

⇒ Karakteri yazdırmak için RC0 'dan başlanarak sırasıyla sütunlardan biri aktif diğerleri pasif yapılır.

⇒ 'E' harfini yazdırmak için önce RC0 pinine lojik 1 gönderilerek ilk sütun aktifleştirilir, diğer pinlere 0 gönderilir. Ardından '111111' sayısı B portuna gönderilerek ilk sütunda istenilen ledlerin yanması sağlanır.

⇒ Sonra kısa bir gecikme yapılır.

⇒ Ardından, RC1 uçu aktif yapılır, diğer uçlar pasif yapılır. B portuna '1001001' sayısı gönderilerek ikinci sütuna ait gerekli ledlerin yanması sağlanır.
⇒ Bu şekilde işlem devam eder.

⇒ Her bir sütuna ait diğer LOOKUP TABLE'a yazılarakda işlem yapılabilir.

⇐ KESMELER ⇒

RB0 KESMESİ

⇒ RB0 pinine dışarıdan uygulanan sinyal ile yükselen veya düşen kenar duştuğunda oluşan kesmedir.

⇒ OPTION ve INTCON registerlarıyla kontrol edilir.

⇒ Option registerının INTEDG biti ile kesmenin yükselen yada düşen kenarda oluşacağı seçilir.

- INTEDG = 1 ise kesme yükselen kenarda,
- INTEDG = 0 ise kesme düşen kenarda oluşur.

⇒ RB0 kesmesini aktif yapmak için INTCON registerının INTE biti "1" yapılmalıdır. Ayrıca GIE biti "1" yapılarak tüm global kesmelere izin verilir.

⇒ Kesmeler aktifleştirildikten sonra, RB0 kesmesi oluştuğunda INTCON'un INTF biti 1 olur. Program kesme alt programına gider.

⇒ RB0 kesmesi için kullanılan registerlar;

- OPTION registeri → INTEDG
- INTCON registeri → GIE, INTE, INTF

RBCHANGE KESMESİ

⇒ RB4, RB5, RB6, RB7 pinlerinde lojik bir değişim gerçekleştiğinde oluşan kesmedir.

⇒ INTCON registeriyla kontrol edilir.

⇒ RBCHANGE kesmesini aktif yapmak için INTCON registerının RBIE biti 1 yapılmalıdır. Ayrıca GIE biti "1" yapılarak tüm global kesmelere izin verilir.

⇒ RBCHANGE kesmesi oluştuğunda INTCON'un RBIF biti 1 olur ve program kesme alt programına dallanır.

⇒ RBCHANGE kesmesinde kullanılan register;
INTCON Registeri → RBIE, RBIF, GIE

⇐ TİMER0 ⇒

- 8 bittir (0 - 256 arası sayar)
- FF'den 0'a dönünce kesme oluşur.
- Zamanlayıcı veya sayıcı olarak kullanılabilir.

⇒ OPTION registerının TOSC biti 0 yapılırsa, Timer0 zamanlayıcı olarak kullanılır. TOSC biti 1 yapılırsa, Timer0 sayıcı olarak kullanılır.

⇒ Prescaler, Watchdog Timer ve Timer0 modülü arasında paylaşılır.
OPTION registerının 3. biti (PSA) bu işlemek kullanılır.

- PSA=0 ise prescaler Timer0 modülüne,
- PSA=1 ise prescaler Watchdog Timer için atanır.

⇒ TMR0 kaydedicisi 00'dan ff'e ulaştıktan sonra tekrar 00'a döndüğünde oluşan taşmadan dolayı kesme meydana gelir.

- INTCON'un 2. biti (T0IF)=1 ise TMR0 kesmesi oluşmuştur. T0IF=0 ise kesme oluşmamıştır.

⇒ TMR0 kesmesini aktif etmek (enable) için;
INTCON'un 5. biti (T0IE)=1 ise TMR0 kesmesi aktiftir. T0IE=0 ise TMR0 kesmesi pasiftir.

⇒ Prescaler değeri OPTION registerının PS2, PS1, PS0 bitleriyle belirlenir.

PS2	PS1	PS0	TMR0 Oranı
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Timer0 Hesaplama ⇒

⇒ Timer0 sayma adımı süresi = T_{KOMUT} X Timer0 oranı

⇒ Kesme süresi = Timer0 sayma adımı süresi X (256 − ÖnDeğer)

Örnek ⇒ Osilatör frekansı = 4MHz.
Prescaler Oranı = 1/8
Timer0 başlangıç değeri = 0
} ise Timer0 kaç Μsn aralıklarla kesme üretir?

$f = 4MHz \Rightarrow T = \frac{1}{4} = 0,25 \mu sn$

$T_{KOMUT} = 4 \times T = 4 \times 0,25 = 1 \mu sn$

Timer0 Kesme Süresi = T_{KOMUT} X Timer0 oranı X (256 − Başlangıç Değer)

$= 0,25 \mu sn \times 8 \times (256 - 0) = 2048 \mu sn$

Örnek = Osilatör frekansı 4MHz dan bir PIC için, OPTION registerındaki PS2, PS1, PS0 bitlerinin değerleri sırasıyla B'111' şeklindedir. TMR0 başlangıç değeri 0 kabul edilirse bu PIC kaç sonra TMR0 kesmesi üretir?

$f = 4MHz \Rightarrow T_{KOMUT} = 1 \mu sn$

Prescaler Oranı B'111' için 1:256 dır.

Kesme Süresi = $1 \mu sn \times 256 \times (256 - 0) = 65536 \mu sn = 0,065 sn.$

Örnek = Osilatör = 4MHz
Prescaler = 1/8
Başlangıç Değeri = 6
Sayaç = 500
} ise kaç sn'de kesme üretilir?

$T_{KOMUT} = 1 \mu sn$

Kesme Süresi = $1 \mu sn \times 8 \times (256 - 6) = 2000 \mu sn$

Gecikme Süresi = $2000 \times 500 = 1000000 \mu sn = 1 sn$
↓
sayaç

Timer0 Kesmesinde kullanılan Registerlar;

- TMR0 → Timer0 registeri
- INTCON registeri → GIE, T0IE, T0IF
- OPTION registeri → T0CS, PS2, PS1, PS0

⇐ TİMER1 ⇒

⇒ 16 bittir
⇒ FFFF'den 0000'a tasma durumunda kesme üretir.
⇒ TMR1H ve TMR1L olmak üzere 8'er bitlik iki kaydedicisi bulunur.
⇒ Timer1, zamanlayıcı ve sayıcı olarak iki ayrı modda kullanılır. Bu mod seçimi T1CON registerının TMR1CS biti ile belirlenir.
- TMR1CS = 1 ise sayıcı modunda
- TMR1CS = 0 ise zamanlayıcı modunda çalışır.

⇒ TMR1 kesmesini aktif etmek için PIE1 registerının 0. biti olan TMR1IE bitini lojik 1 yapmak gerekir. Pasif yapmak isterse TMR1IE bitini lojik 0 yapmak gerekir.

⇒ Timer1 modülü, T1CON registerının 0. bitinin (TMR1ON) durumuna göre açık/kapalı konumuna getirilebilir.

⇒ Timer1 prescaler oranları ⇒
Prescaler oranları T1CON registerının T1CKPS1, T1CKPS2 bitleriyle ayarlanır.

T1CKPS1	T1CKPS2	Prescaler Oranı
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

Timer1 Hesaplaması;

Kesme Süresi = T_{komut} × Timer1 Prescaler Oranı × (65536 − TMR1 Başlangıç Değeri)

Örnek ⇒ Osilatör frekansı = 8MHz
Prescaler Oranı = 1:8
Sayaç = 5

} ise bu Timer1 kesmesinin 1sn gecikme oluşturabilmesi için gerekli başlangıç değerini bulunuz.

$F = 8MHz \Rightarrow T = \frac{1}{8} = 0,125\,\mu sn \Rightarrow T_{KOMUT} = 0,5\,\mu sn$

$1000000\,\mu sn = 0,5 \times (65536 - Başlangıç) \times 8 \times 5$

$65536 - Başlangıç = 5000$

Başlangıç Değer = 15536 = H'3CB0'

TMR1H = 3C
TMR1L = B0

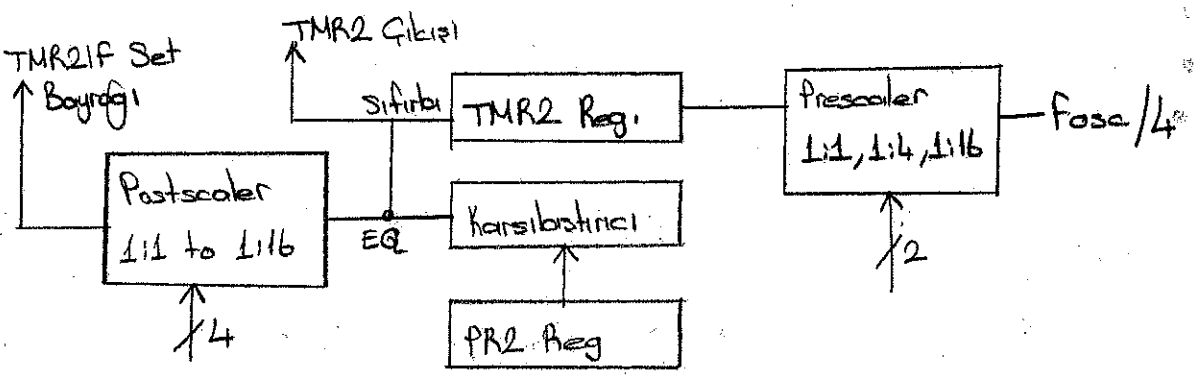
Timer1 kullanılan Registerlar;

- INTCON Registeri → GIE, PEIE
- PIE1 Registeri → TMR1IE
- TMR1L Registeri
- TMR1H Registeri
- T1CON Registeri → T1CKPS1, T1CKPS0, TMR1ON
- PIR1 Registeri → TMR1IF

← TİMER2 ⇒

⇒ 8 bittir.
⇒ 8 bitlik PR2 kaydedicisini kullanır.
⇒ TMR2 kaydedicisi, PR2 kaydedicisine eşitse kesme üretir.
⇒ Prescaler ve postscaler oranlarına bağlı kesme üretir.

⇒



⇒ Prescaler değerlerini ayarlamak için T2CON registerının T2CKPS1, T2CKPS0 bitleri kullanılır.

T2CKPS1	T2CKPS0	Prescaler
0	0	1:1
0	1	1:4
1	X	1:16

⇒ Timer2 postscaler değerleri;

T2CON → TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	Oran
0	0	0	0	1:1
0	0	0	1	1:2
0	0	1	0	1:3
⋮	⋮	⋮	⋮	⋮
1	1	1	1	1:16

⇒ Timer2 'nin 8 bitlik PR2 kaydedicisi vardır. Timer2 modülü 00'dan başlayıp içindeki değer PR2 'ye eşit olduğunda kesme üretir ve 00'a düşer.

⇒ Timer2 modülü kullanılmadığında, TMR2ON (T2CON'un 2. biti) bitinin "0" yapılmasıyla kapatılabilir.

Timer2 kesmesinde kullanılan registerlar;

- INTCON registeri → GIE, PEIE
- PIE1 registeri → TMR2IE
- T2CON registeri → TOUTPS3, TOUTPS2, TOUTPS1, TOUTPS0 → postscaler
- T2CON registeri → TMR2ON
- T2CON registeri → T2CKPS1, T2CKPS0 → prescaler
- PR2 registeri
- PIR1 registeri → TMR2IF

← PWM →

- PWM modunda, CCP2 (RC2) pininden 10 bitlik PWM çıkışı alınır.
- PWM modunda işlem yapabilmek için RC2 pini çıkış yapılmalıdır.
- PWM modunu ayarlamak için CCP1CON, CCPR1L ve T2CON registerları kullanılır.

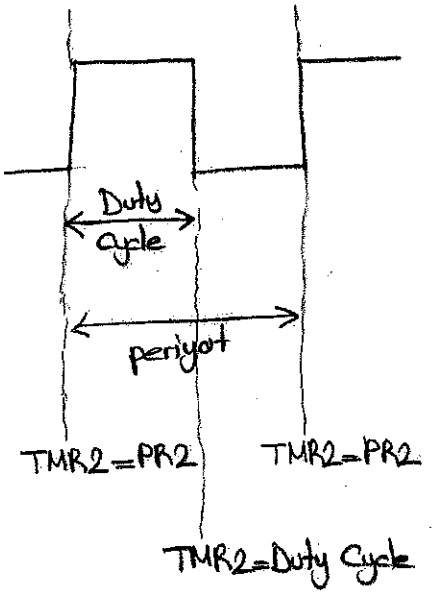
⇒ CCP1CON ⇒

—	—	CCPxX	CCPx4	CCPxM3	CCPxM2	CCPxM1	CCPxM0
Bit7							Bit0

CCPxX, CCPx4 ⇒ PWM 'in en az değerlikli bitleri, pals dalulük oranının en az değerliksiz 2 bitidir.

CCPxM3, CCPxM2, CCPxM1, CCPxM0 ⇒ CCPx modülü için işlem modu seçme bitleridir.

- PWM modunu seçmek için CCP1CON registerının CCPM3 ve CCPM2 bitleri "1" yapılmalıdır.
- 10 bitlik PWM çözünürlük verisinin yüksek değerlikli 8 biti CCPR1L 'de, düşük değerlikli 2 biti CCP1CON 'un (CCPxX, CCPx4) 5 ve 4. bitlerinde bulunur.



TMR2 = PR2 olduğu anda PWM sinyali lojik 1 olur. TMR2 = Duty Cycle olduğu anda ise PWM sinyali lojik 0'a düşer.

- En kısa süreli duty cycle süresi, osilatörün bir periyotluk süresi ile aynıdır. Örneğin, Osilatör 10MHz ise T=0,1 Mşn' dir. Duty Cycle oranları 0,1 Mşn' dir.
- Maksimum duty cycle için seçilebilecek en büyük çözünürlük değeri 2^{10} (1024)' dir.

• Eğer PWM duty cycle süresi, PWM periyodundan daha uzunsa, RC2 (ccp1) pini hiçbirzaman "0" olmaz.

• PWM duty cycle süresini belirleyeceğimiz kaydedicilerin (ccp1L, ccp1X, ccp1Y) değeri 0 seçilirse, RC2 çıkışı hep ON olur.

⇒ <u>PWM Periyodu</u> ⇒ TMR2 = PR2 olana kadar geçen süredir.

PWM Periyodu = [(PR2)+1] • 4 • Tosc • (TMR2 prescaler değeri)

⇒ <u>PWM frekansı</u> ⇒ Birim zamanda oluşabilecek sayım sayısıdır. TMR2'nin prescaler değeri değiştirilerek PWM frekansı aralığı belirlenebilir. Yüksek frekanslarda PWM çözünürlüğü düşer.

$$\text{PWM frekansı} = \frac{1}{\text{PWM Periyodu}}$$

⇒ <u>PWM Duty Cycle</u> ⇒ Sinyalin lojik1'den lojik0'a geçmesi için geçen süredir. Yani RC2 girişinin lojik1'de kalma süresidir.

PWM Duty Cycle = (ccpr1L : ccp1con<5:4>) • Tosc • (TMR2 prescaler değeri)

••• Not = TMR2 = PR2 olduğu durumda bir sonraki periyota kadar şunlar olusur;
→ TMR2 kaydedicisi temizlenir.
→ ccp1 pini lojik1 olur.
→ PWM duty cycle süresi ccpr1L'den ccpr1H'a atılır.

⇒ <u>PWM Çözünürlüğü</u> ⇒ Darbe genişliğinin %0'dan %100'e kadar kaç eşit aralıkta değiştirilebileceğini belirten değerdir. Bu değer en az 2 bit, en fazla 10 bit ile ifade edilebilir.

• 2 bitlik bir PWM çözünürlüğü darbe genişliğini %25, %50, %75 ve %100 olarak 4 farklı süre için ayarlanabileceğini belirtir.
• PWM frekansı ile PWM çözünürlüğü ters orantılıdır.

$$\text{PWM Çözünürlüğü} = \frac{\log\left(\frac{F_{osc}}{f_{PWM}}\right)}{\log(2)} \text{ bit}$$

⇒ PWM işleminin yapılması için adımlar;

- PWM periyodunun belirlenmesi için, PR2 kaydedicisine gerekli değer yazılır.
- PWM duty cycle süresinin belirlenmesi için, CCPR1L kaydedicisine ve CCP1CON <5:4> bitlerine gerekli değerler yazılır.
- RC2 pini çıkış yapılır.
- TMR2 prescaler değeri ayarlanır ve T2CON tarafından TMR2 yetkilendirilir.
- CCP1 modülü PWM işlemi için düzenlenir. (CCP1CON kaydedicisinin CCPM3 ve CCPM2 bitleri 1 yapılır.

⇒ PWM İşlemi için kullanılan registerlar;

- CCP1CON → CCPxX, CCPx4, CCPxM3, CCPxM2, CCPxM1, CCPxM0
- PR2 kaydedicisi
- CCPR1L kaydedicisi
- T2CON kaydedicisi

Örnek ⇒ $f_{osc} = 20MHz$, TMR2 prescaler = 1:1 olduğunda, PWM frekansının 78,125 KHz olması için PR2 kaydedicisinin değeri kaç olmalıdır?

$$\text{PWM frekansı} = [(PR2)+1] \cdot 4 \cdot T_{osc} \cdot 1$$

$$\frac{1}{78.125KHz} = [(PR2)+1] \cdot 4 \cdot \frac{1}{20MHz} \cdot 1$$

$$12.8\mu sn = [(PR2)+1] \cdot 0,2\mu sn \cdot 1 \Rightarrow PR2 = 63$$

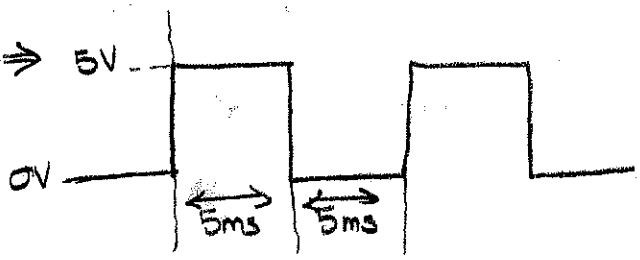
Örnek ⇒ 78.125KHz frekans ve 20MHz osilatör frekansı kullanılırsa PWM çözünürlüğü?

$$\frac{1}{78.125KHz} = 2^{\text{PWM çöz.}} \cdot \frac{1}{20}MHz \cdot 1 \Rightarrow 12.8ms = 2^{\text{PWM çöz.}} \cdot 50ns \cdot 1$$

$$256 = 2^{\text{PWM çöz.}} \Rightarrow \log(256) = (\text{PWM çöz.}) \log(2) \Rightarrow \text{PWM çöz.} = 8 \text{ bit.}$$

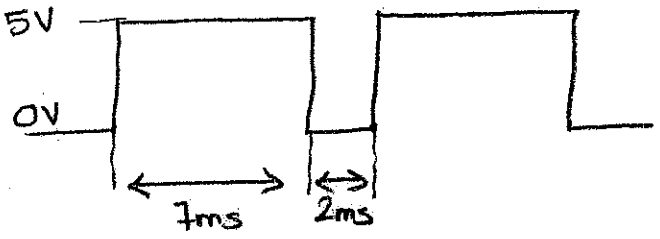
- Bu durumda darbe genişliği %0'dan %100'e kadar $2^8 = 256$ eşit aralıkla ayarlanabilir.
- Daha yüksek çözünürlüklere ulaşmak için PWM frekansı azaltılmalıdır.

Örnek ⇒



Görev periyodu (Duty Cycle) = %50

5V . %50 = 2,5V → 2,5V lojik 1 uygulanmış gibidir.



Görev Periyodu = %70

5V . %70 = 3,5V lojik 1 uygulanmış gibidir

- Ledde sürekli 3,5V uygulamakla 7ms 5V, 3ms 0V uygulamak aynıdır.

⇒Örnek = ⊗ led1 → f = 100Hz, δ = %70 (duty)
⊗ led2 → f = 200Hz, δ = %70
Buna göre hangi led parlak yanar.

- Frekans, saniyede yanıp sönme sayısıdır. Parlaklıkla ilgili değildir.
- Duty cycle ise lojik 1'de kalma süresidir. Ledlerin duty cycle süreleri aynı olduğundan parlaklıklarında aynıdır.

Örnek ⇒ PWM frekansı 5kHz, duty cycle %30 olan led ne kadar süre lojik 1, ne kadar süre lojik 0 durumunda kalır, PR2 değeri kaçtır? (Fosc = 4MHz, Prescaler = 4)

PWM frekansı = 5kHz ⇒ PWM periyodu = $\frac{1}{5kHz} = 0,2 msn = 200 \mu sn$

- Duty Cycle Oranı %30 ise PWM periyodunun %30'u kadar süre led lojik 1 durumundadır (led yanık)

lojik 1 süresi = $200 \times \frac{30}{100} = 60 \mu sn$ → led yanık

lojik 0 süresi = $200 \times \frac{70}{100} = 140 \mu sn$ → led sönük

- PWM Periyodu = [PR2+1] x 4 x Tosc x TMR2 Prescaler = 200 µsn

Fosc = 4MHz ⇒ Tosc = $\frac{1}{4} = 0,25 \mu sn$

[PR2+1] x 4 x 0,25 µsn x 4 = 200 µsn PR2 = 49

Duty Cycle = [CCPR1L : CCP1CON<5:4>] x Tosc x TMR2 Prescaler

60 µsn = [CCPR1L : CCP1CON<5:4>] x 0,25 µsn x 4

60 = CCPR1L : CCP1CON<5:4>

↓

B'00001111 00' = CCPR1L : CCP1CON<5:4>

CCPR1L CCP1CON<5:4>

Örnek → Fpwm = 4 kHz , PWM Duty Cycle Oranı = %50 , Fosc = 8 MHz, Prescaler 4 ise CCPR1L ve CCP1CON<5:4> değerlerini bulunuz.

- PWM Periyodu = $\frac{1}{4\,kHz}$ = 0,25 msn = 250 µsn

  Tosc = $\frac{1}{Fosc} = \frac{1}{8\,MHz}$ = 0,125 µsn

  PWM Periyodu = [PR2 + 1] x 4 x Tosc x Prescaler = 250 µsn

  [PR2 + 1] x 4 x 0,125 x 4 = 250 µsn

  PR2 = 124

- Lojik1 süresi (Duty Cycle) = 250 x $\frac{50}{100}$ = 125 µsn

  Lojik0 süresi = 125 µsn

- Duty Cycle = [CCPR1L : CCP1CON<5:4>] x Tosc x Prescaler = 125 µsn

  [CCPR1L : CCP1CON<5:4>] x 0,125 µsn x 4 = 125 µsn

  CCPR1L : CCP1CON<5:4> = 250 = B'00111110 10'

  CCPR1L CCP1CON<5:4>

NOT⇒ PWM Modunda;

T2CON ⇒ B'000000--'

- postscaler (kullanılmaz)
- TMR2ON
- prescaler → prescaler = 00 → 1:1, 01 → 1:4, 1x → 1:16

CCP1CON ⇒ 00 -- 1100

- 00 → kullanılmaz
- 11 → PWM aktif
- 00 → kullanılmaz
- Duty Cycle ağırlıksız bitler

Not⇒ PWM Modunda;

- RC2 çıkış yap.
- CCP1CON'a 00--1100 değerini yükle
- CCPR1L'yi sıfırla
- PR2'ye gerekli değeri yükle
- PWM_Değer değişkenine değer yükle
- T2CON'a 00000100 değerini yükle.

COMPARE Modunda;

- PIE1'in CCP1IE bitini aktifleştirerek CCP kesmesine izin ver
- INTCON'un GIE ve PEIE bitlerini aktifleştir.
- CCPR1L ve CCPR1H'a gerekli değerleri yükle
- CCP1CON'a kesme olacağı zaman yapılacak işlemi belirt ⇒ B'0000 ----'
- T1CON'a B'00--0001" değerini yükle.

CAPTURE Modunda;

- RC2 veri girişi yapılır.
- PIE1'in CCP1IE biti ile kesmelere izin verilir.
- INTCON'un PEIE ve GIE bitleri "1" yapılır.
- CCP1CON'a gerekli değer verilerek kaç yükselen kenarda bir işlem yapılacağı seçilir.
- TMR1 için gerekli değerler verilir.

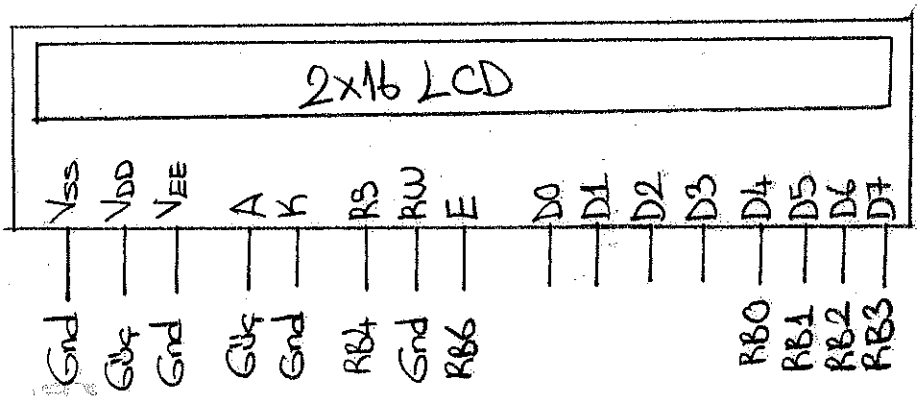
NOT⇒ PWM ve Compare modunda RC2 çıkış, Captue modunda giriş yapılır.

⇐ LCD ⇒

⇒ Işıksız lcd'ler 14 pinli, ışıklı lcd'ler ise 16 pinlidir. Bu pinlerden 8'i veri, 3'ü kontrol, 2'si besleme, 1'i kontrast, 2 pinde arkaplan aydınlatması için kullanılır.

⇒ 8 adet veri yolunun hepsi kullanılabileceği gibi, yalnızca 4 taneside (4 bitlik iletişim) kullanılabilir. 4 bitlik kullanmanın avantajı, mikrodenetleyicinin fazla sayıda pinini meşgul etmemektir.

⇒



⇒ 4 bitlik iletişim için pin bağlantıları

⇒ Pin Açıklama

Vss —— Gnd
VDD —— 5V (besleme)
VEE —— Kontrast ayar pini
RS —— Kaydedici seçimi
RW —— Okuma/yazma seçimi
E —— Yetkilendirme
D0 —— Veri biti - 0
D1 —— " - 1
D2 —— " - 2
D3 —— " - 3
D4 —— " - 4
D5 —— " - 5
D6 —— " - 6
D7 —— " - 7
A —— Aydınlatma Besleme
K —— Aydınlatma Gnd

⇒ VEE pini;

Potansiyometreye bağlanarak kontrast ayarının yapılmasında kullanılabilir. Eğer bu pin kullanılmayacaksa toprağa (gnd) bağlanır.

⇒ RS pini;

RS pini 0 yapılırsa LCD'ye komut gönderildiği, 1 yapılırsa karakter gönderildiği anlaşılır.

⇒ RW pini;

RW pini 0 yapılırsa LCD'ye veri transferi yapılır, 1 yapılırsa LCD'den veri alımı gerçekleşir.

⇒ E pini;

E pini, LCD ve mikrodenetleyici arasında, komutların veya karakter verilerinin gerçek anlamda aktarımını başlatmak için kullanılır. LCD'ye veri yazılırken, veri aktarımı sadece sinyalin düşen kenarında gerçekleşir. LCD'den okuma yapılırken ise veri yükselen kenarın hemen ardından hazır olur ve sinyal tekrar düşmeye kadar hatta kalır.

⇒ LCD komutları;

Komut	Hex
• Ekranı sil	01
• Display ve imleç başa	02,03
• Karakter giriş modu	04-07
• Display on/off ve imleç	08-0F
• Display/imleç kaydır	10-1F
• fonksiyon ayarları	20-3F
• CGRAM adres ayarı	40-7F
• Display adres ayarı	80-FF

⇒ Önemli komutlar;

- 02,03 → imleci başa konumlandır.
- 28 → 2 satır, 5x8 dot matris, 4 bit iletişim
- 38 → 2 satır, 5x8 dot matris, 8 bit iletişim
- 0C → ekran açık, altçizgi yok, imleç yanıp sönme durumu yok.
- C0 → 2. satıra geç, 1C → sağa kaydır, 04 → ters yazı yaz

ADC

- ADC modülü, analog bir değeri mikrodenetleyicinin işleyebileceği şekilde ikilik sayı sistemine yani dijitale dönüştürür.
- Elde edilen dijital sonuç ADRESH ve ADRESL registerlarına yazılır.
- 16F877A mikrosistemlerinde ADC modülü 10 bittir. Yani analog bilgi dijitale çevrildiğinde oluşan dijital veri 10 bittir.
- 16F877A mikrosistemlerinde 8 tane ADC kanalı vardır. Hangi kanal seçilirse o kanaldaki analog veri dijitale çevrilir.
- ADC modülünün analog girişlerinde CHOLD adında bir tutma kondansatörü vardır. Bu kondansatör analog girişten gelen değerin belirli bir süre sabit tutulmasını sağlayarak daha doğru dijital sonuç oluşturulmasını sağlar.
- ADC modülü 4 tane kaydediciye sahiptir;

  ADRESH → ADC modülünün yüksek seviyeli sonuç kaydedicisi
  ADRESL → ADC modülünün düşük seviyeli sonuç kaydedicisi
  ADCON0 → ADC modülünün kontrol kaydedicisi 0
  ADCON1 → ADC modülünün kontrol kaydedicisi 1

- ADC modülünde kanal seçimi ADCON0 registerının 5,4,3 nolu pinleri (CHS2, CHS1, CHS0) ile yapılır.

Kanal	Port	CHS2	CHS1	CHS0
Kanal 0	RA0/AN0	0	0	0
Kanal 1	RA1/AN1	0	0	1
Kanal 2	RA2/AN2/Vref-	0	1	0
Kanal 3	RA3/AN3/Vref+	0	1	1
Kanal 4	RA5/AN4	1	0	0
Kanal 5	RE0/AN5	1	0	1
Kanal 6	RE1/AN6	1	1	0
Kanal 7	RE2/AN7	1	1	1

- Seçilen kanala ait TRIS biti giriş olarak ayarlanmalıdır.

- ADCON0 →

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
Bit7							Bit0

ADCS1, ADCS0 → A/D çevirici için clock frekansı seçim bitleridir.

CHS2, CHS1, CHS0 → Analog çevirici için analog kanal seçim bitleridir.

GO/DONE → A/D dönüştürücü durum bitidir.

ADON → A/D çeviriciyi yetkilendirme bitidir. (enable/disable)

- ADCON1 →

ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0

ADFM → A/D dönüştürme işlemi sonucunda oluşan dijital sonucun ADRESH, ADRESL kaydedicilerine yerleşim biçimini belirleyen bittir.

ADCS2 → A/D çevirici için clock frekansı seçim bitidir.

PCFG3, PCFG2, PCFG1, PCFG0 → A/D çevirici portunun işlevini belirlemeye yarayan bitlerdir.

••• A/D dönüştürücü için saat frekansı seçimi ADCON0 registerının ADCS1, ADCS0 ve ADCON1 registerının ADCS2 bitleriyle yapılır.

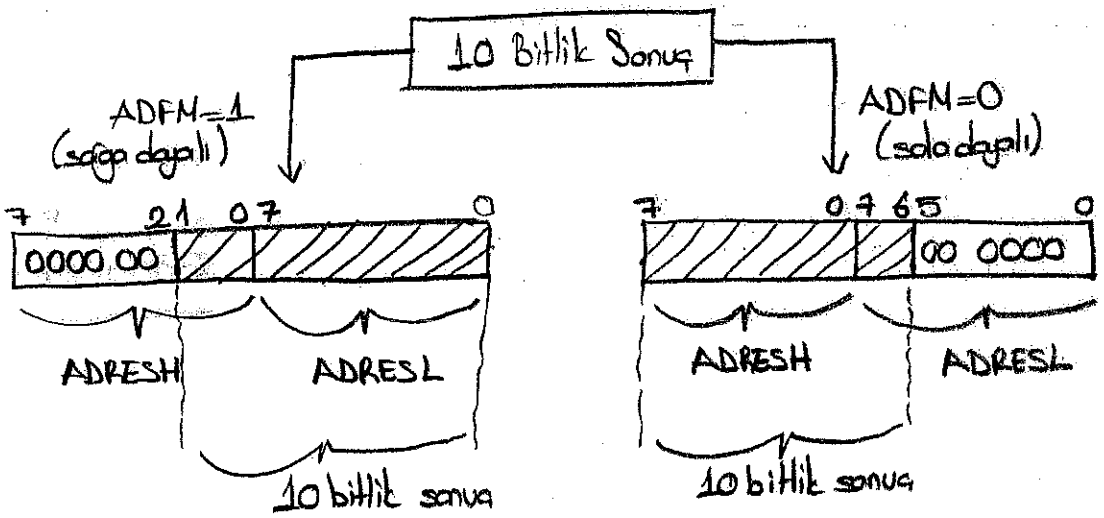
ADCS2, ADCS1, ADCS0	Clock
0 0 0	Fosc/2
0 0 1	fosc/8
0 1 0	fosc/32
0 1 1	FRC (Dahili RC osilatörden gelen clock)
1 0 0	fosc/4
1 0 1	fosc/16
1 1 0	fosc/64
1 1 1	FRC (Dahili RC osilatörden gelen clock)

• GO/DONE biti 0 iken dönüştürücü hiçbir işlem yapmıyor demektir. Dönüştürme işlemini başlatmak için GO/DONE biti "1" yapılır. Dönüştürme işlemi bittiği anda bu bit tekrar 0'a döner.

••• A/D çevirici portunun işlevini belirlemek için ADCON1 registerının PCFG3, PCFG2, PCFG1, PCFG0 bitleri kullanılır.

PCFG3, PCFG2, PCFG1, PCFG0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A
1000	A	A	A	A	V_{ref+}	V_{ref-}	A	A

••• A/D dönüşüm işlemi sonucunda elde edilen 10 bitlik veri ADFM bitinin aldığı değere göre ADRESH, ADRESL registerlarına yerleşir.



• A/D Dönüştürme İşlemi;

Kullanıcı tarafından GO/DONE biti "1" yapıldığında A/D dönüşüm işlemi yaklaşık 100 ns'lik bir gecikme sonrasında başlar. Bu işleme başlarken, CHOLD (tutma) kondansatörünün analog girişle bağlantısı A/D dönüşüm işlemi bitene kadar kesilir. Dönüşüm sonucunda 10 bitlik dijital veri elde edilir. Sonuç ADRESH ve ADRESL kaydedicilerine yazılır.

Bu işlemin sonucunda GO/DONE biti resetlenir ve ADIF bayrağı set edilir. CHOLD kondansatörünün analog girişle bağlantısı donanım tarafından tekrar sağlanır.

⇒ Belirlenmiş doğruluğu elde etmek için A/D çeviricinin CHOLD (tutma) kondansatörü giriş gerilimine şarj olması beklenmelidir.

⇒ Analog kaynaklar için (potansiyometre ...) tavsiye edilen maksimum empedans 10kΩ'dur. (Empedans, alternatif akıma karşı koyan zorluktur.)

⇒ Mikrodenetleyicinin analog sinyali minimum algılama zamanı 19.72 µsn'dir.

TACQ = TAMP + TC + TCOFF

- TCOFF → Sıcaklık katsayısı
- TC → CHOLD şarj zamanı
- TAMP → Yükseltme gecikme zamanı

⇒ A/D Dönüşümü için Saat Kaynağı Seçimi;

- TAD, bit başına A/D dönüşüm zamanıdır.
- 10 bit A/D dönüşümü için maksimum 12TAD periyodu gerekir.
- TAD seçimi için 7 seçenek vardır.

2TOSC	32TOSC
4TOSC	64TOSC
8TOSC	Dahili A/D modülü RC Osilatörü (2-6 µsn)
16TOSC	

- A/D dönüşümünün doğru olarak yapılabilmesi için, TAD zamanı minimum 1,6 µsn olmalıdır.
- TAD zamanı için maksimum işlem frekansları;

A/D Saat Kaynağı (TAD)		Maksimum İşlem
İşlem	ADCS2, ADCS1, ADCS0	frekansı
2TOSC	000	1,25 MHz.
4TOSC	100	2,5 MHz
8TOSC	001	5
16TOSC	101	10
32TOSC	010	20
64TOSC	110	20
RC(1,2,3)	x11	

→ RC kaynağı için TAD zamanı nominalde 4 µsn'dir, fakat 2 ile 6 µsn arasında olabilir.

⇒ A/D Dönüştürme İşleminin Adımları;

1→ ADCON0 ve ADCON1 'e gerekli değerler verilerek analog pinlerin işlevi belirlenir, kanal seçimi, dönüşüm için saat frekansı seçimi yapılır.

2→ ADCON0 'ın ADON biti "1" yapılarak A/D dönüştürücü enable edilir.

3→ Kesme kullanılacaksa PIE1 registerının ADIE biti "1" yapılır.
INTCON registerının GIE ve PEIE bitleri "1" yapılarak kesmeler aktif edilir.

4→ Dönüştürme işlemine başlamadan önce yeterli kadar beklenmelidir.

5→ GO/DONE biti "1" yapılarak dönüşüm başlatılır.

6→ Dönüştürmenin tamamlandığının anlaşılması için GO/DONE bitinin "0" olması kontrol edilerek beklenir. Eğer kesme kullanılıyorsa dönüşüm bittiğini bildiren kesmenin gelmesi beklenir.

7→ Dönüşüm sonucu ADRESH ve ADRESL kaydedicilerinden okunur.

8→ Kesmeden çıkarken ADIF bayrağı sıfırlanır. Ve GO biti yeni dönüşümler için tekrar "1" yapılır.

⇒ A/D modülü ile ilgili kaydediciler;

- INTCON → GIE, PEIE
- PIE1 → ADIE
- PIR1 → ADIF
- ADRESH, ADRESL Registerları
- ADCON0 → ADCS1, ADCS0, CHS2, CHS1, CHS0, GO/$\overline{\text{DONE}}$, ADON
- ADCON1 → ADFM, ADCS2, PCFG3, PCFG2, PCFG1, PCFG0

NOT⇒ Sıcaklık Sensörü;

- Algıladığı sıcaklığı belirli bir değerle çarparak çıkışa verir.
- Celcius tabanlı ölçüm yapar.
- Her 1°C'lik sıcaklık değişiminde çıkış 10mV değişmektedir.
- Besleme gerilimi 4-30V arasındadır.
- 5V ile beslendiğinde −55° ile 150°C arasında ölçüm yapılır.

⇒ Transfer fonksiyonu=

- A/D dönüsüm sonucunda oluşacak digital bilginin herbir adımı;

  Referans Voltajı /1024

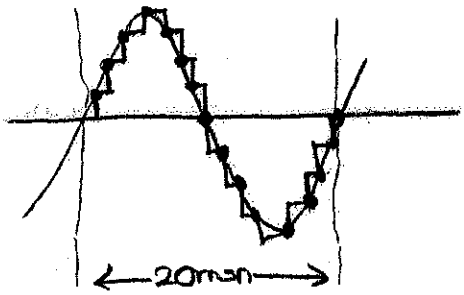
  formülü ile bulunur.

- Dolayısıyla 10 bitlik bir ADC'de 1024 farklı ikilik çıkış durumu oluşur,
- Çözünürlük (resolution), digital bilginin bit sayısı olarak tanımlanır.

⇒ Referans voltajı 5V kabul edersek;

5 Volt /1024 = 0,00488... = 4,88 mV → digital her 4,88 mV'de bir artar.

⇒



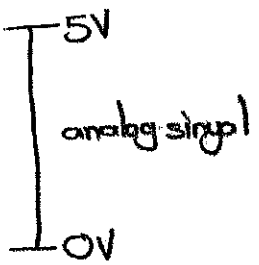
Analog sinyali digitale çevirdikten sonra, bu sinyali tekrar analoga çevirirsek, başlangıçtaki analog sinyali elde edemeyiz. En yakın analog sinyali elde edebilmek için örnekleme sayısı arttırılmalıdır.

⇒ 50 tane örnek alırsak;

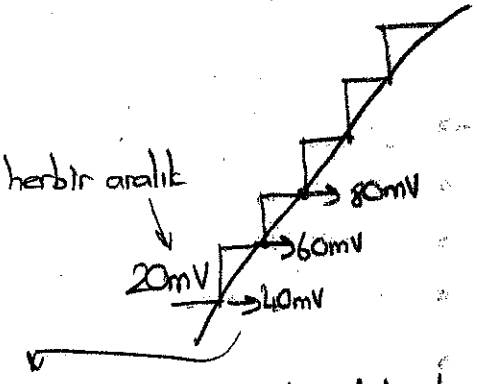
20ms /50 = 400 µsn' de bir örnek almış oluruz.

- Herbir örnekteki değeri registera kaydedersek (8 bitlik ADC'de) en fazla 256 farklı değer kaydedilebilir.

⇒



5V | 256
20mV → herbir aralık



- Çözünürlük 20mV ise 20mV'luk aralıktaki değerler muhtemelen aynıdır.
- 40mV ile 60mV aralığındaki tüm değerler aynı digital sonuca çevirilir.

Örnek ⇒ Kullandığımız PIC'teki ADC 10 bit olduğundan $2^{10}=1024$ farklı dijital sonuç elde edilir.

Analog giriş gerilimleri 0V-5V ise;

$$5V \;|\; \frac{1024}{4,8\,mV \cong 5mV}$$

0mV – 5mV arası ⇒ 0000000000
5mV – 10mV arası ⇒ 0000000001
!

Örnek ⇒ $V_{REF+}=12V$, $V_{REF-}=-12V$ } Buna göre 0V analog değerin dijital karşılığı nedir?

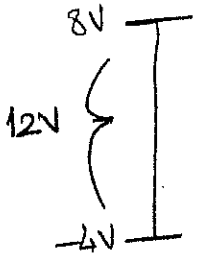
ADC 10 bitlik ise ⇒ 2^{10} = 1024 (adet/nurluk)

$$24V \;|\; \frac{1024}{0,023V}$$

-12V ile -11,977V arası ⇒ 0000000000
-11,977V ile -11,954V " ⇒ 0000000001
!
0V, -12V ile 12V'un tam ortası olduğundan 512 olur.

Örnek ⇒ $V_{REF+}=8V$, $V_{REF-}=-4V$, ADC, 8 bit ise

- -1,57V analog değerin karşılığı=?
- 5,57V " " "
- 8,5V " " "



8 bit ADC = 2^8 = 256 (adet/nurluk)

$$12V \;|\; \frac{256}{0,046V}$$

-4V ile -1,57V arasında 2,43V vardır.

$$\frac{2,43}{0,046}=52$$

Örnek ⇒ ADRESH = D'20', ADRESL = D'30' olsun. Buna göre ADFM=1 ve ADFM=0 iken digital sonuç ne dur?

ADRESH → 00101000
ADRESL → 00011110

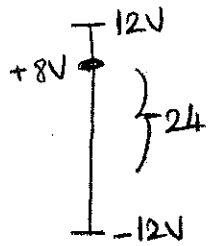
ADFM=0 ⇒ B'0010100000' = D'160' = H'0A0' → sola dayalı

ADFM=1 ⇒ B'0000011110' = D'30' = H'01E' → sağa dayalı

Örnek (Quiz Sorusu) ⇒

$V_{REF+} = 12V$
$V_{REF-} = -12V$
ADC, 10 bit olduğuna göre 8V'a karşılık gelen değeri bulunuz ve ADRESH, ADRESL kaydedicilerinin içeriğini yazınız.

Çözünürlük $= 2^{10} = 1024$



$\frac{24}{1024} = 0,023V$

-12V ile 8V arasında 20V var

$\frac{20}{0,023} = 869,56$

Yani 8V'a karşılık gelen değer 869'dur. B'1101100101'

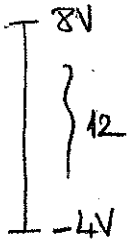
ADFM=0 sola dayalı ⇒ 1101100101,000000
(AdresH, AdresL)
digital sonuç

Örnek ⇒ $V_{REF+} = 8V$

$V_{REF-} = -4V$

ADC 8 bit ise $-1,57$ Volt'un digital karşılığı nedir?

Çözünürlük $= 2^8 = 256$



$\frac{12}{256} = 0,046\ V$

$-4V$ ile $-1,57V$ arasında 2,43 fark varsa;

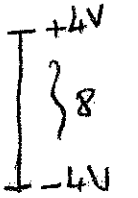
$\frac{2,43}{0,046} = 52,82$ → Yani $-1,57V$ 'un digital karşılığı 52'dir.

Örnek ⇒ $V_{REF+} = +4V$

$V_{REF-} = -4V$

ADC 3 bit ise $-2,5V$ 'un digital karşılığı nedir?

Çözünürlük $= 2^3 = 8$



$\frac{8}{8} = 1V$

$-4V$ ile $-2,5V$ arasında 1,5 fark varsa;

$\frac{1,5}{1V} = 1,5$ → yani $-2,5V$ 'un digital karşılığı D'1'dir.

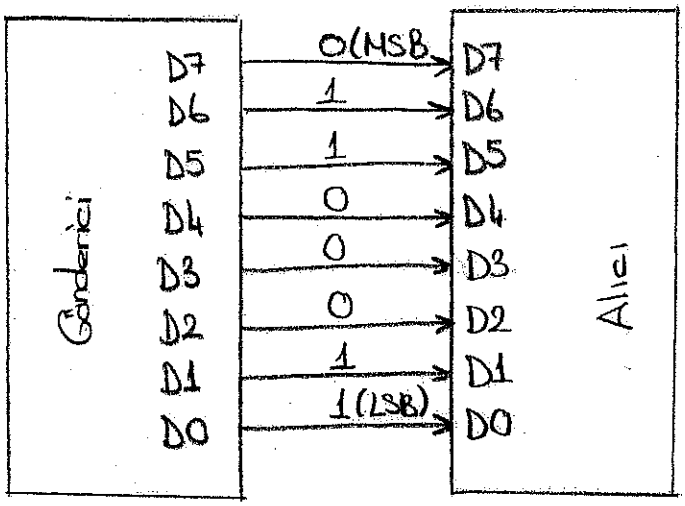
Sağlama ⇒ $-4V$ ile $-3V$ arası → 000
$-3V$ ile $-2V$ arası → 001 → $-2,5V$ bu aralıkta olduğundan digital değeri "1" dir.
$-2V$ ile $-1V$ arası → 010
$-1V$ ile $0V$ arası → 011
$0V$ ile $1V$ arası → 100
$1V$ ile $2V$ arası → 101
$2V$ ile $3V$ arası → 110
$3V$ ile $4V$ arası → 111

SERİ İLETİŞİM

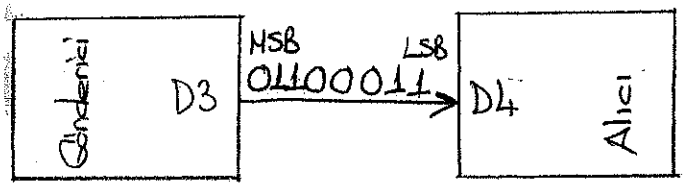
Paralel – Seri Haberleşme

⇒ B'11000110' verisini karşı tarafa gönderme;

Paralel iletişim;



Seri iletişim;



- Paralel iletişimde birden fazla kanal vardır. Gönderilecek verinin her bir biti aynı anda bu kanallardan gönderilir. Seri iletişime göre daha hızlıdır.

- Seri iletişimde tek kanal vardır. Gönderilecek verinin bitleri LSB bitinden başlanarak sırasıyla gönderilir. Paralel iletişimden daha fazla zaman alır.

⇐ USART MODÜLÜ ⇒

Baud Rate Generator ⇒ (BRG)

- Baud Rate Generatörü (BRG), Usart modülünde saat palsının frekansını belirler, yani Usart modülünün hem senkron hemde asenkron modunda çalışmasındaki sinyal hızını belirler.

- Asenkron modunda, BRGH (TXSTA<2>) biti ile sinyal hızı kontrol edilir. Senkron modunda kullanılmaz.
  BRGH=0 ise düşük hız, BRGH=1 ise yüksek hız seçilir.
- BRG'nin kontrolü SPBRG register ile yapılır.

- Baud Rate ⇒

- Baud Rate, saniyede gönderilen bit sayısıdır yani iletişim hızı olarak ifade edilir.

- Baud Rate Hesabı;

BRGH=0 (düşük hızda) ⇒ $\text{Baud Rate} = \frac{\text{Fosc (Hertz)}}{64(x+1)}$

BRGH=1 (yüksek hızda) ⇒ $\text{Baud Rate} = \frac{\text{Fosc (Hertz)}}{16(x+1)}$

- Buradaki x değerine tablodan bakılarak karşılık gelen değer SPBRG registerına yazılır.

Baud Rate	SPBRG (Fosc=4)	SPBRG (Fosc=20MHz)	
2400	D'25'	D'129'	
9600	D'6'	D'31'	⇐ BRGH=0
19200	D'2'	D'15'	
2400	D'103'	—	
9600	D'25'	D'129'	⇐ BRGH=1
19200	D'12'	D'64'	

⇒ Usart Modülünün kontrolü için RCSTA ve TXSTA registerları kullanılır.

- TXSTA →

CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
Bit7							Bit0

CSRC ⇒ Clock kaynağı seçim bitidir. Asenkronda kullanılmaz.
TX9 ⇒ 9 bit'lik veri gönderme yetkilendirme bitidir.
TXEN ⇒ Göndermeyi aktif yapma bitidir.
SYNC ⇒ Senkron/Asenkron modunu seçme bitidir. (0-Asenkron)
BRGH ⇒ Yüksek/Düşük baud rate seçim bitidir.
TRMT ⇒ Transmit shift register (TSR) durum bitidir.
TX9D ⇒ 9 bit göndermede 9. bit buraya yazılır.

- RCSTA ⇒

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
Bit7							Bit0

* SPEN ⇒ Seri port yetkilendirme bitidir.
* RX9 ⇒ 9 bitlik veri almayı yetkilendirme bitidir.
SREN ⇒ Tek veri alımı yetkilendirme bitidir. Asenkron modda kullanılmaz.
* CREN ⇒ Sürekli veri almayı yetkilendirme bitidir.
ADDEN ⇒ Adres algılamayı etkinleştirme bitidir.
FERR ⇒ Çerçeve hata bitidir.
* OERR ⇒ Gönderme shift register durum bitidir.
* RX9D ⇒ 9 bitlik veri almada 9. bit buraya yazılır.

NOT ⇒ Usart modunda gönderilirken ve alınırken, alınan ve gönderilen ilk bit herzaman LSB bitidir.

NOT ⇒ Asenkron modda işlemi durdurmak için PIC uyku moduna geçirilir.

⇒ RX → giriş
TX → çıkış
} Genel bağlantı

BSF TRISC,7 → RX
BCF TRISC,6 → TX

⇒ Usart ile Asenkron Veri Gönderme;

• Veri Gönderme İşlemi;

o Gönderilecek veri TXREG kaydedicisine yüklenir. TXREG kaydedicisine yüklenen veriler donanım tarafından TSR'ye transfer edilir. Gönderilecek olan veriler TSR'ye yüklendikten sonra veri iletimi başlar, TXREG'in içeriği TSR'ye yüklendiğinde TXIF bayrağı kalkar.

o TXREG'in içeriği TSR'ye yüklendiğinde yeni bir veri TXREG kaydedicisine yüklenebilir.

o TXIF bayrağını yazılımla sıfırlamak mümkün değildir. Bu bayrak sadece TXREG kaydedicisine yeni bir veri yüklendiğinde otomatik olarak sıfırlanır.

• TSR kaydedicisindeki verinin TX/RC6 pini ile gönderilmesi bittiğinde TRMT biti "1" olur. Bu durum TSR'nin boşaldığını gösterir.

o Veri TXREG'den TSR'ye yüklendiğinde TXIF bayrağı "1" olup kesme oluşur ancak verinin TSR'den karşı tarafa gönderme işleminin bittiğini belirten bir kesme yoktur. TSR'nin boş olup olmadığını anlamak için TRMT biti kontrol edilir.

o Verinin Gönderme İşleminde İzlediği yol;

TXREG ⟶ TSR ⟶ RC6/TX pini → Alıcı

o Asenkron Veri Gönderimi Yapılırken Yapılması Gereken İşlemler;

— Uygun baud rate için SPBRG'ye gerekli değer verilir.
— Yüksek veya düşük hız seçmek için (TXSTA<2>) BRGH bitine gerekli değer verilir.
— Asenkron çalışma modunu seçmek için TXSTA<4> biti olan SYNC'ye "0" değeri verilir.
— RC6/TX ve RC7/RX pinlerini seri port pinleri olarak ayarlamak için RCSTA<7> biti olan SPEN bitine "1" değeri verilir.
— Göndermeyi aktif duruma getirmek için TXSTA<5> biti TXEN'e "1" verilir.
— 8 bit veri gönderimini seçmek için TXSTA<6> biti TX9 lojik 0 yapılır.

— Kesme kullanılacaksa PIE1 registerındaki TXIE biti "1" yapılarak TX kesmesi aktif duruma getirilir.
— Kesmeleri aktif duruma getirmek için INTCON'un GIE ve PEIE bitleri "1" yapılır.

NOT⇒ Eğer 9 bit'lik veri gönderimi yapılmak isteniyorsa;

- TXSTA'nın 6. biti olan TX9 biti "1" yapılarak 9 bit veri gönderimi seçilir.
- 9. bit TXSTA'nın TX9D bitine yazılır.

●●● USART ile Asenkron Veri Gönderiminde Kullanılan Registerlar;

— INTCON ⇒ GIE, PEIE
— PIE1 ⇒ TXIE
— PIR1 ⇒ TXIF
— TXSTA ⇒ TX9, TXEN, SYNC, BRGH, TRMT, TX9D
— RCSTA ⇒ SPEN
— TXREG kaydedicisi
— SPBRG kaydedicisi

⇒ Usart ile Asenkron Veri Alma;

••• Veri Alma İşlemi;

• RC7/RX pini aracılığıyla alınan veri RSR (Receive Shift Register)'ye transfer edilir. RSR'den alınan veri RCREG kaydedicisine gönderilir. Verinin RSR'den RCREG'e transferi tamamlandığı anda RCIF bayrağı kalkar. Ardından veri RCREG kaydedicisinden alınıp istenilen işlem yapılır.

• RCIF bayrağı sadece okunabilirdir. Yazılımla sıfırlamak mümkün değildir. Donanım RCIF bayrağını RCREG kaydedicisi okunurken ve boş olduğunda otomatik sıfırlar.

• RCREG kaydedicisi çift FIFO (ilk giren ilk çıkar) yapıdadır. Çift FIFO, ilk gelen veri ön tamponda tutulurken, yeni bir veri geldiğinde eskisi arka tampona itilir. Çift FIFO ile yazılım tarafından veri okumada gecikme olduğunda, hatalı veri okumaya önlem alınmış olur.

• Veri Alma İşleminde Verinin İzlediği Yol;

RC7/RX ⟶ RSR(Receive Shift Register) ⟶ RCREG

• Asenkron Veri Alımı Yapılırken Yapılması Gereken İşlemler;

— Uygun baud rate için SPBRG'ye gerekli değer verilir.
— Yüksek veya düşük hız seçmek için BRGH (TXSTA<2>) bitine 0-1 değeri verilir.
— Asenkron çalışma modunu seçmek için TXSTA<4> biti olan SYNC'ye "0" değeri verilir.
— RC6/TX ve RC7/RX pinlerini seri port olarak ayarlamak için RCSTA<7> biti olan SPEN bitine "1" değeri verilir.
— Veri alımını aktif etmek için RCSTA<4> biti olan CREN biti "1" yapılır.
— 8 bitlik veri alımını seçmek için RCSTA<6> biti RX9'a "0" değeri verilir.
— Eğer kesme kullanılacaksa PIE1'in RCIE bitи "1" yapılır.
— Kesmeleri aktif etmek için INTCON'un GIE ve PEIE bitleri "1" yapılır.

NOT⇒ 9 bitlik veri alımı için;
RCSTA'nın RX9 biti "1" yapılarak 9 bitlik veri alımı seçilir.
9. bit RCSTA'nın RX9D bitine yazılır.

USART ile Veri Almada Kullanılan Registerlar

- INTCON → GIE, PEIE
- PIE1 → RCIE
- PIR1 → RCIF
- RCSTA → SPEN, RX9, CREN, OERR, RX9D
- TXSTA → SYNC, BRGH
- RCREG Registeri
- SPBRG Registeri

⇐ Hesaplama Örnekleri ⇒

1→ İstenilen baud rate 9600 bps ise X değeri ve hata oranını bulunuz?

$$\text{İstenen Baud Oranı} = \frac{Fosc}{64(X+1)} \qquad (\text{Düşük hız} \rightarrow BRGH = 0)$$

Fosc = 20MHz.

$$9600 = \frac{20000000}{64(X+1)} \Rightarrow X = 31,552 = 31 \rightarrow \text{Tablodan bakılır. (SPBRG değeri)}$$

$$\text{Hesaplanan Baud Oranı} = \frac{20000000}{64(31+1)} = 9766 \text{ bps}$$

$$\text{Hata} = \frac{\text{Hesaplanan} - \text{İstenen}}{\text{İstenen}} = \frac{9766-9600}{9600} = 0,0173 = 1,73\%$$

2→ İletişim hızı 2400 bps ise 1sn'de kaç karakter gönderilir?

1 sn'de 2400 bit ⇒ 2400 bps olduğundan

Her karakter 8 bit olduğundan

$$\frac{2400}{8} = 300 \text{ karakter}$$

3→ 2400 bps hızla 1KBytlık veri kaç sn'de iletilir?

2400 bps = 1 sn'de iletilen bit sayısıdır.

1 KByte = 1024 x 8 bit = 8192 bit

1 sn'de	2400 bit
X	8192 bit

X = 3,41 sn

4→ 2,5 sn'de 2400 bps hızla kaç byte veri gönderilir?

1 sn'de	2400 bit
2,5 sn'de	X

[illegible] 75 byte

5→ Bir komutu 0,5 µsn'de icra eden bir mikroişlemci 1 bit veriyi 1600 µsn'de gönderebilmesi için düşük hızda SPBRG registirine hangi değer yüklenmelidir?

$T_{KOMUT} = 0,5\,\mu sn \Rightarrow T_{osc} = T_{KOMUT} \times 4 \Rightarrow T_{osc} = \frac{0,5\,\mu sn}{4} = 0,125\,\mu sn$

$f_{osc} = \frac{1}{T_{osc}} = \frac{1}{0,125} = 8\,MHz.$

1 bit	1600 µsn
y	1000000 µsn

$y = 625$ bps

Düşük hızda BRGH = 0 ise;

$Baud\ Rate = \frac{f_{osc}}{64(x+1)} \Rightarrow 625 = \frac{8000000}{64(x+1)} \quad x = 199$