ADuC 841 µ-denetleyicisi – (2)

Y. Müh. Ayhan Yüksel

(Son güncelleme: 07.03.2012 - Zafer İşcan)

Tıbbi Enstrumantasyon Tasarım & Uygulamaları

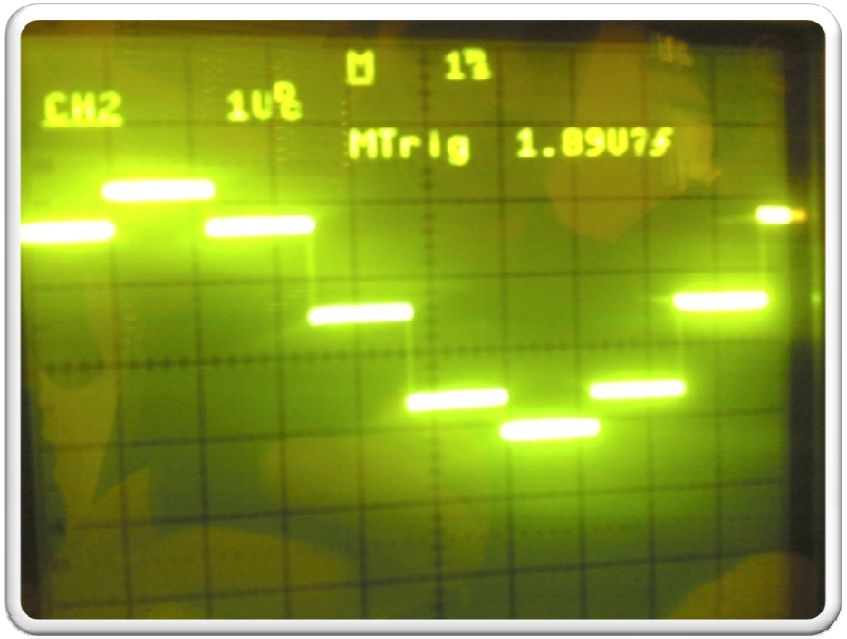
(07.03.2012)

Sunum Planı

- **Mikrodenetleyici Çevre Elemanları**
  - ADC (Analog – Sayısal Çevirici)
  - DAC (Sayısal – Analog Çevirici)
  - Timer (Zamanlayıcı)
  - UART (Evrensel Asenkron Alıcı/Verici)

- **Kesmeler (Interrupts)**

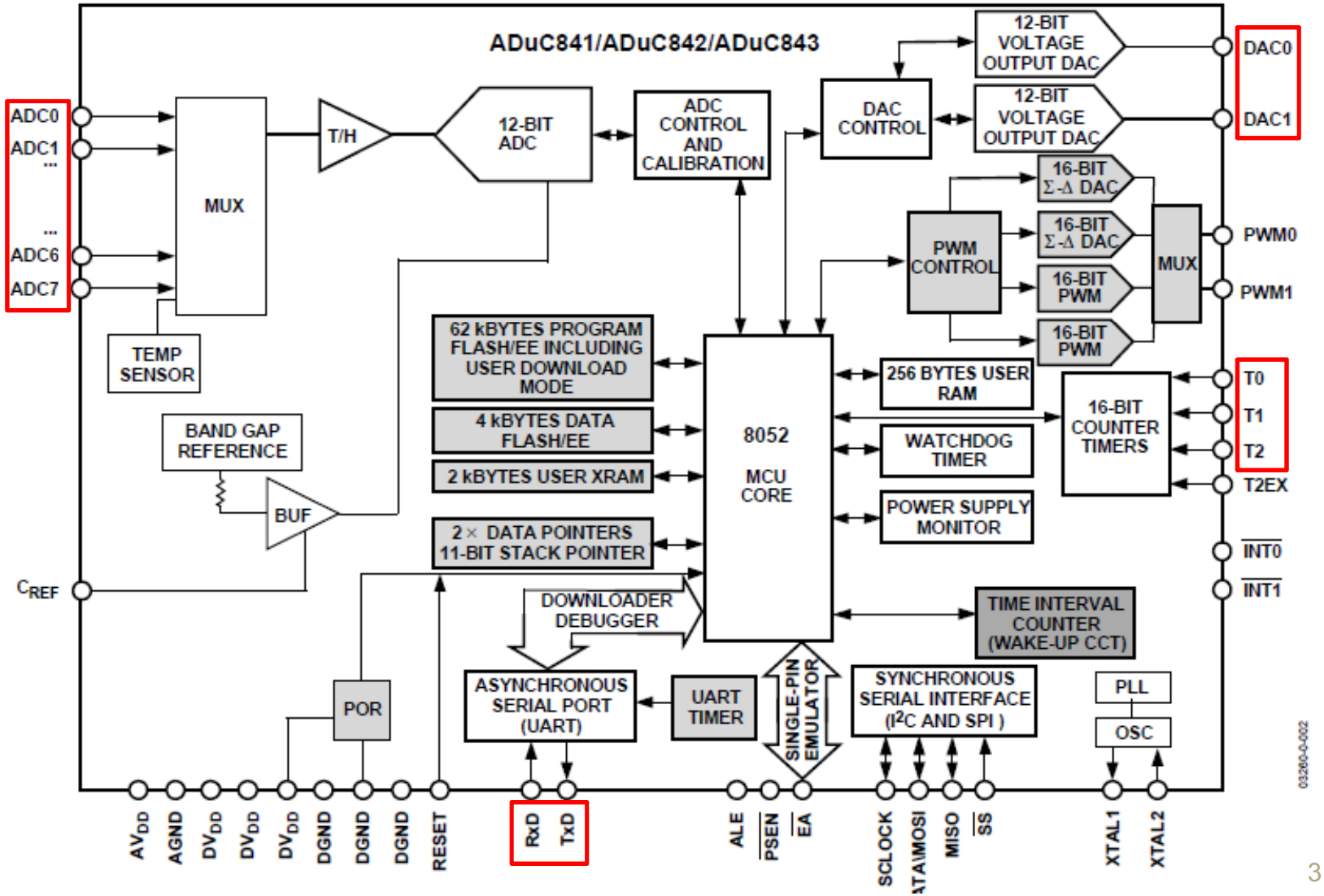
- **Port Kullanımı**



(18.03.2009 Uygulama 2)

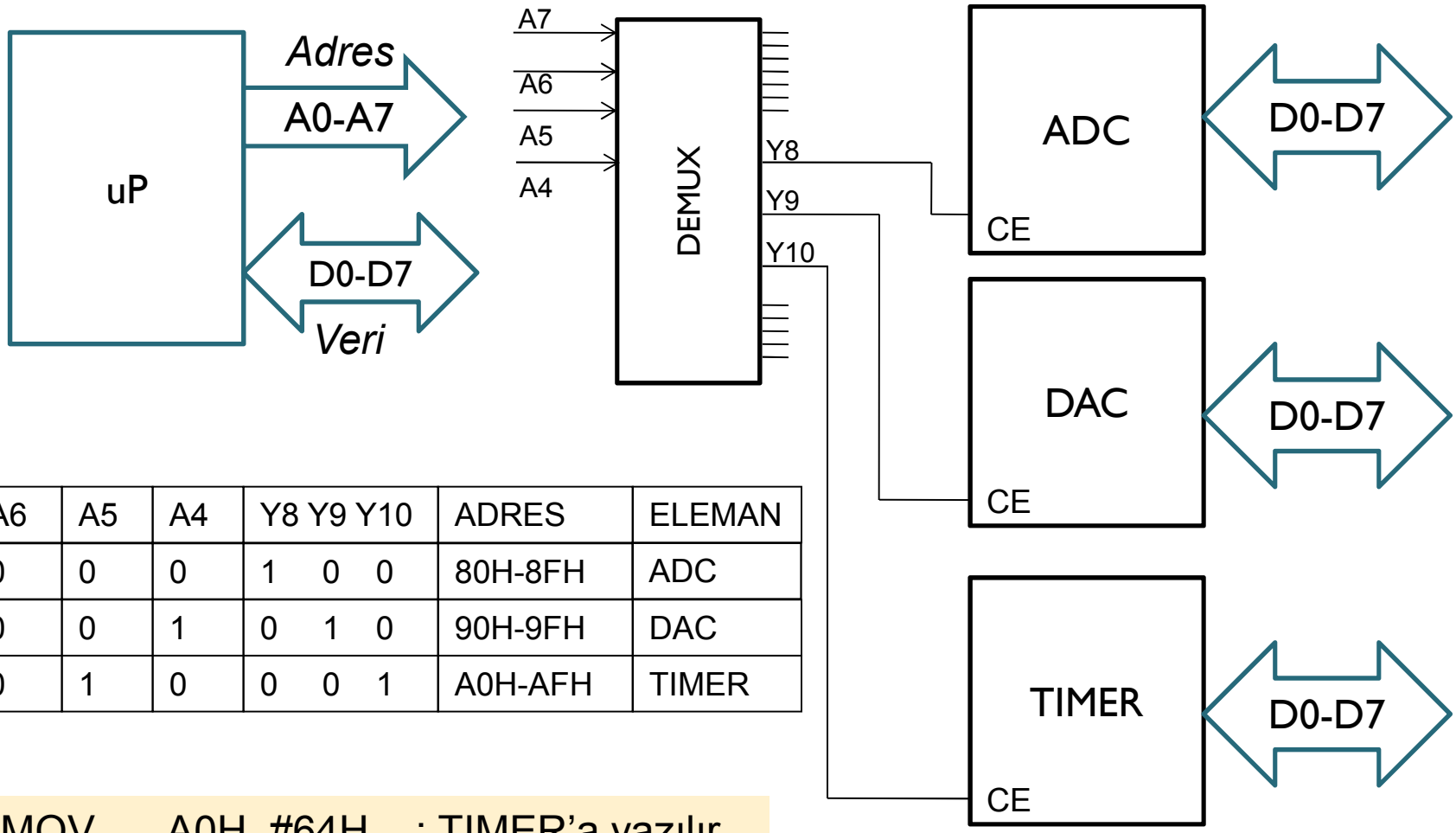
MİKRODENETLEYİCİ ÇEVRE ELEMANLARI

- Bir mikrodenetleyici, bir mikroişlemci ile beraber etrafında bulunan çeşitli donanımlardan oluşur.
- Bu donanımlar yardımıyla, mikrodenetleyici dış ortamla iletişim kurabilir.



MİKRODENETLEYİCİ ÇEVRE ELEMANLARI

- Her donanımın bir adresi (bazen birden fazla) bulunmaktadır. Böylece istenilen donanım, o donanımın adresine bir bilgi yazılarak kontrol edilebilir.



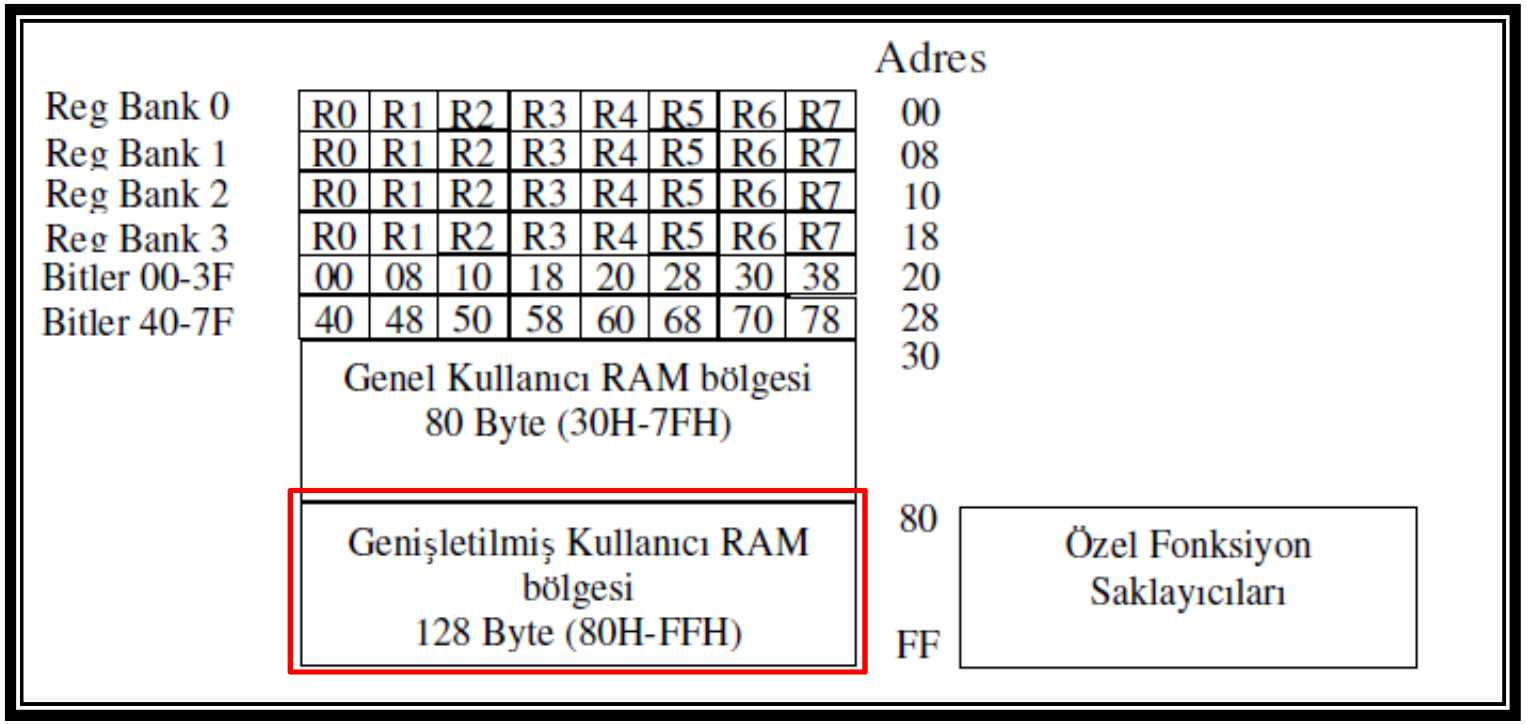
A7	A6	A5	A4	Y8 Y9 Y10	ADRES	ELEMAN
1	0	0	0	1 0 0	80H-8FH	ADC
1	0	0	1	0 1 0	90H-9FH	DAC
1	0	1	0	0 0 1	A0H-AFH	TIMER

```
MOV     A0H, #64H    ; TIMER'a yazılır
MOV     A,     80H    ; ADC okunur
```

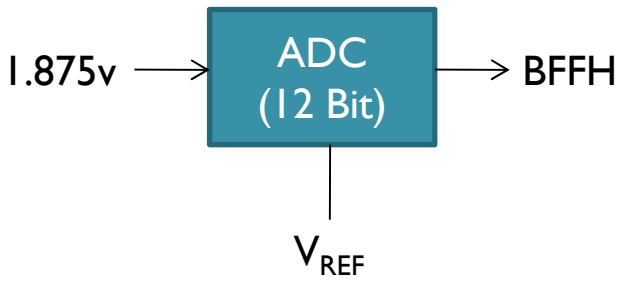
CE: Chip Enable

SFR (Special Function Register) - Özel Fonksiyon Saklayıcısı

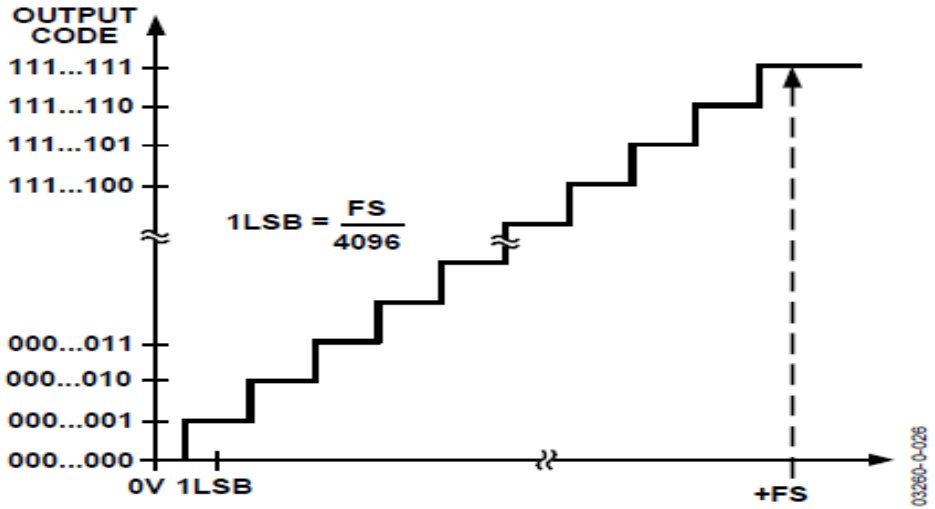
- ADuC çipi içindeki çeşitli donanımları kontrol etmek, bu donanımlara veri yazmak ya da bu donanımlardan veri okumak için kullanılırlar.
- SFR'lerin belirli adresleri vardır.
- SFR'ler dahili belleğin üst 128 byte'lık kısmında bulunurlar. Doğrudan adresleme ile erişilirler.
- SFR'lerden bir kısmı bit adreslenebilir olabilir.



ADC (Analog sayısal çevirici)

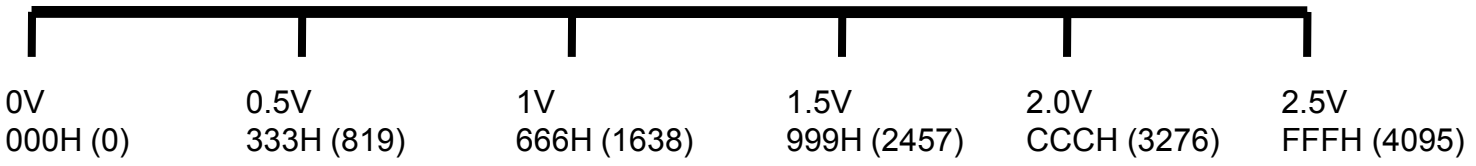


- ADC, dış ortamdaki analog gerilim değerlerini sayısal koda dönüştüren bir çevre birimidir.
- ADuC 841'de 12 bit çözünürlüğünde 8 adet ADC bulunmaktadır.
- 0V ile V_{REF} arası 12 bitlik değere dönüştürülür.
- V_{REF} ayarlanabilirdir (normal durumda 2.5V).



Çıkış Kodu = $(FFFH).\frac{V_{in}}{V_{REF}}$

V_{REF} =2.5V iken:



ADC’lerin ayarlanması ve kullanılması

- Diğer donanımlar gibi ADC de SFR’ler ile ayarlanır ve kullanılır.
- **ADCCON1**, **ADCCON2**, **ADCCON3** saklayıcılarına uygun değerler yazılmasıyla, ADC’ler kullanılabilir hale gelmektedir.
- ADCCON1 ve ADCCON2’nin ayarlanması gerekirken, ADCCON3 kalibrasyon yapmak amacıyla kullanılır.

ADCCON1—(ADC Control SFR 1)

ADCCON1 saklayıcısı ADC'lerin okuma hızını ayarlar, kullanıma sokar ya da kapatır. Bu saklayıcı ile, okuma frekansı ve bir okuma için harcanması gereken süre belirtilmektedir.

Bit	İsim	Açıklama
7	**MD1**	ADC mod biti. ADC'ler çalıştırılacak ise 1, kapatılacaksa 0 yapılır.
6	**EXT_REF**	Referans geriliminin nereden alınacağını bildirir. Harici referans için 1, dahili referans için 0 kullanılır.
5	**CK1**	5. ve 4. bitler ADC saatini (CLK) ayarlar. 00 seçilirse, master CLK 32'ye, 11 seçilirse 2'ye bölünür. ADC saatinin 8 MHz'in altında olması önerilir.
4	**CK0**	
3	**AQ1**	3. ve 2. bitler analog sinyalin tutulma süresini belirler. 11 seçilirse 4 (daha yavaş), 00 seçilirse 1 ADC saati (daha hızlı) tutulur. (10 ya da 11 değeri tavsiye edilmektedir)
2	**AQ0**	
1	**T2C**	1 seçilirse ADC'yi TIMER2'ye bağlar, böylece ADC'nin TIMER2 taşma kesmesiyle okumaya başlayacağını belirtir. Örnekleme frekansı da TIMER2 tarafından belirlenir.
0	**EXC**	Okuma işlemi için harici bir tetik kullanılacağını bildirir.

ÖRNEK: ADC'ler dahili referans gerilimi ile çalışacaktır. ADC çevrimleri TIMER2'den bağımsız olacaktır. ADC'ler Okuma işleminin doğruluğu için düşük hızda çalıştırılacaktır. Bu amaçla ADCCON1 saklayıcısı **8CH** olarak aşağıdaki gibi ayarlanır.

```
                              7 6 5 4   3 2 1 0  - Bit No
MOV    ADCCON1,#8CH ;         1 0 0 0   1 1 0 0
```

ADCCON2—(ADC Control SFR 2)

ADCCON2 saklayıcısı ile kanal ve okuma mod'u seçimi yapılır.
Aşağıda ADCCON2 saklayıcı yapısı gösterilmektedir.

Bit	İsim	Açıklama
7	**ADCI**	ADC kesme biti. Donanım tarafından tek bir ADC çevriminde veya DMA blok çevrimi neticesinde set edilir (1 yapılır).
6	**DMA**	DMA modu etkinleştirme biti
5	**CCONV**	Sürekli çevrim etkinleştirme biti
4	**SCONV**	Tek çevrim etkinleştirme biti. 1 yapılarak ADC'nin bir veri okuması sağlanır.
3	**CS3**	3,2,1,0 nolu bitler kanal seçimini göstermektedir. Hangi kanaldan okuma yapılacağı belirlenir.
2	**CS2**	
1	**CS1**	
0	**CS0**	

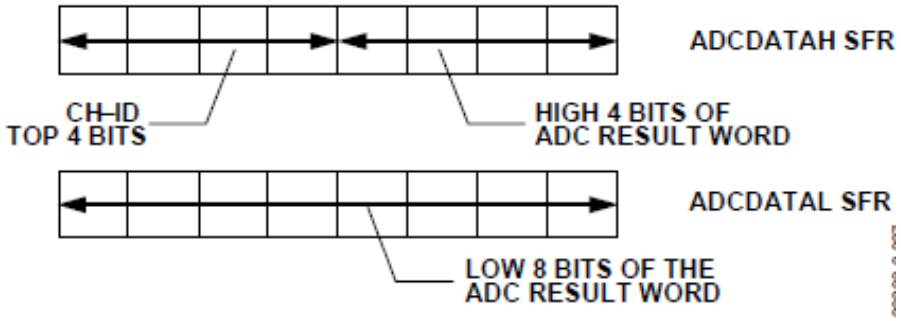
ÖRNEK: ADC kullanılarak 0. kanaldaki gerilim değeri bir defalığına okunmak üzere ayarlanacaktır. Gerekli kodu yazın.

```
MOV     ADCCON1, #8CH ;      1 0 0 0  1 1 0 0   <-- ADC ayarlama
MOV     ADCCON2, #00H  ;     0 0 0 0  0 0 0 0   <-- ADC kanal ayarlama
SETB    SCONV                                   <-- ADC okumaya başlama
```

ADC verisini okuma

12 bit ADC okuma sonucu 2 SFR'ye yazılır (**ADCDATAH, ADCDATAL**) .

Yüksek anlamlı bitleri içeren SFR (ADCDATAH) ayrıca, okunan kanalın numarasını da içerir.

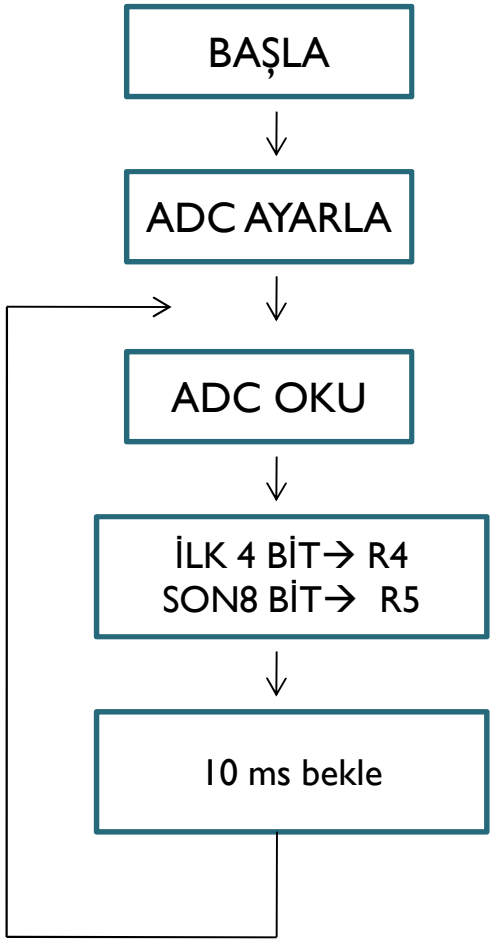


ÖRNEK: Kanal 1 üzerindeki analog gerilim değeri (1.875V) ADCDATAL ve ADCDATAH saklayıcılarına atandığında, bu saklayıcılar aşağıdaki değerlere sahip olacaktır:

1.875V→BFFH

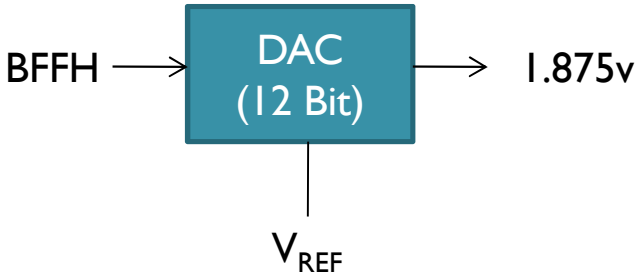
ADCDATAH : 1BH (0001 1011)
ADCDATAL : FFH (1111 1111)

ÖRNEK: ADC girişindeki gerilim değeri her 10 ms'de bir okunacak ve okuma değerleri R4-R5 saklayıcılarına atılacaktır. Gerekli kodu yazınız.

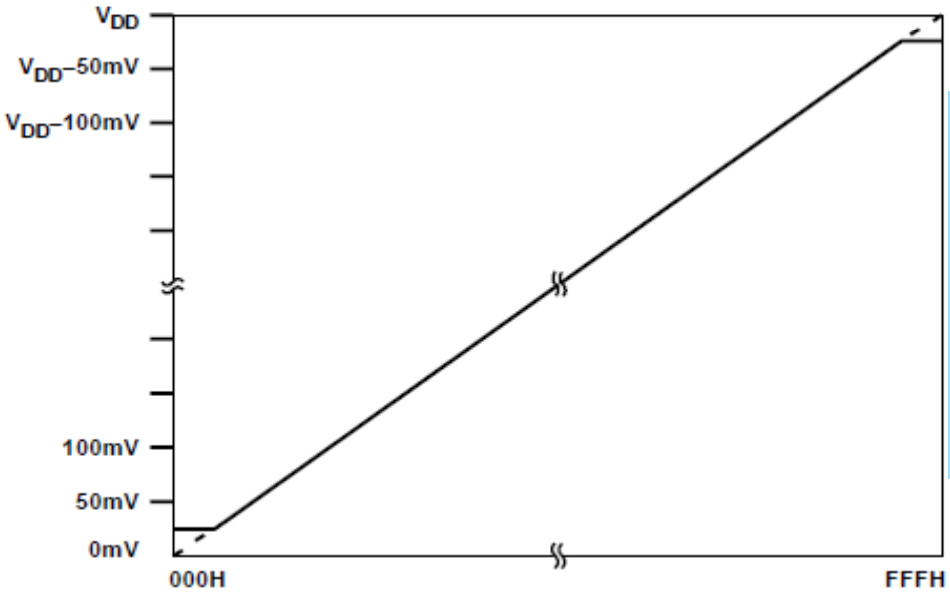



```
$MOD841         ; use 8052 predefined symbols
CSEG
ORG 0000H       ;start code from 0000H
BASLA:
MOV  ADCCON1, #8CH  ; 1 0 0 0   1 1 0 0
MOV  ADCCON2, #01H  ; 0 0 0 0   0 0 0 1
OKU:
SETB   SCONV    ;ADC'den bir veri oku
JNB    ADCI,$   ;çevrim bitene kadar bekle
CLR    ADCI
MOV    R4,ADCDATAH
MOV    R5,ADCDATAL
ANL    04H,#0FH ;04H adresinde R4 var.
MOV    A, #01H  ;gecikme için kullanılacak
CALL   DELAY    ;gecikme fonksiyonunu çağır
JMP    OKU
;_____________________________________________
DELAY:
           MOV     R5,A
DLY2:      MOV     R7,#090h
DLY1:      MOV     R6,#0FFh
           DJNZ    R6,$
           DJNZ    R7,DLY1
           DJNZ    R5,DLY2
           RET
END
```

DAC (Sayısal Analog Çevirici)



- DAC, sayısal veriyi analog gerilim değerine dönüştüren bir çeviricidir.
- ADuC 841'de 12 bit çözünürlüğünde 2 adet DAC bulunmaktadır.
- 12 bitlik değer 0V ile V_{DD} arasına dönüştürülebilir.
- Ayrıca 0-V_{REF} arasında da kullanılabilir.



Örnek:
VDD=3.3V için çıkışa 1.5V yazılmak isteniyor:

4095*(1.5/3.3)=1861=0745H

Çıkışa 0745H kelimesi yazılmalıdır.

DAC kullanma

- Sayısal analog çevirici, DAC kontrol saklayıcısına (**DACCON**) gerekli değer yazılarak ayarlanır.
- DAC'ların düzgün çalışması için ADC'ler etkin olmalıdır.

DACCON DAC Control Register

Bit	İsim	Açıklama
7	**MODE**	Veri tipinin 8 bit ya da 12 bit olarak ayarlanmasını sağlar (1: 8 bit, 0: 12 bit).
6	**RNG1**	DAC1 için gerilim üst sınırını belirler (0: 0-VREF, 1: 0-VDD).
5	**RNG0**	DAC0 için gerilim üst sınırını belirler (0: 0-VREF, 1: 0-VDD).
4	**CLR1**	1 değeri DAC1 çıkışını normal değerinde bırakır, 0 yazılırsa çıkışı 0V olmaya zorlar.
3	**CLR0**	1 değeri DAC0 çıkışını normal değerinde bırakır, 0 yazılırsa çıkışı 0V olmaya zorlar.
2	**SYNC**	1 seçildiğinde, çıkış gerilimi DACxL'ye yazıldığı anda değiştirilir. 0 iken çıkış gerilimini değiştirmek için SYNC biti 1 yapılmalıdır.
1	**PD1**	DAC1 aktif/kapalı biti (1: aktif, 0:kapalı)
0	**PD0**	DAC0 aktif/kapalı biti (1: aktif, 0:kapalı)

ÖRNEK: DAC0, 12 bitlik datayı çıkışa verecektir. Çıkış gerilimi sınırı 0-V_{DD} arasında olacaktır. Senkronizasyon, gerilim değerinin düşük anlamlı saklayıcısına (DACL) bir değer yazıldığı zaman olacaktır. DACCON saklayıcısına uygun değeri yazınız.

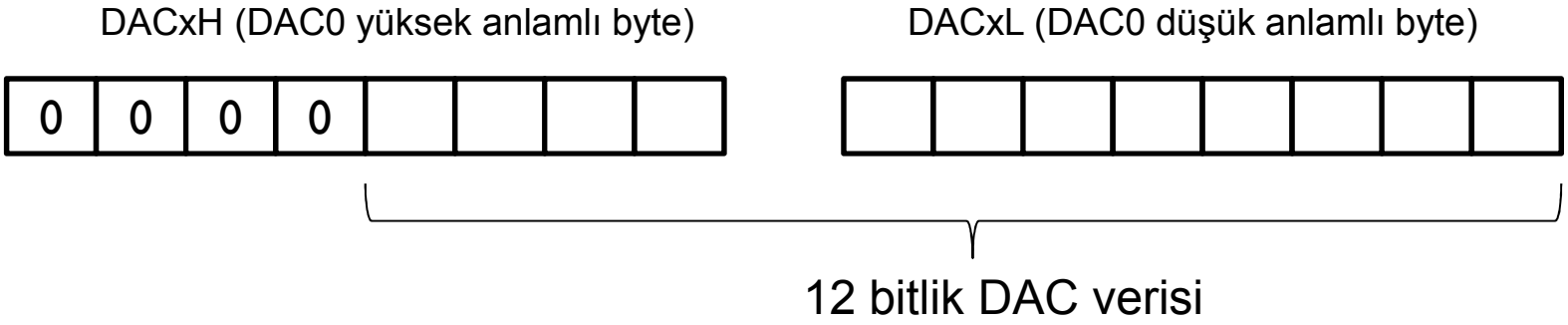
```
MOV     ADCCON1, #80H
MOV     DACCON,  #2DH
```

DAC kullanma

DACxH/L **DAC Data Registers**

DAC0L (DAC0 düşük anlamlı byte) DAC0H (DAC0 yüksek anlamlı byte)

DAC1L (DAC1 düşük anlamlı byte) DAC1H (DAC1 yüksek anlamlı byte)



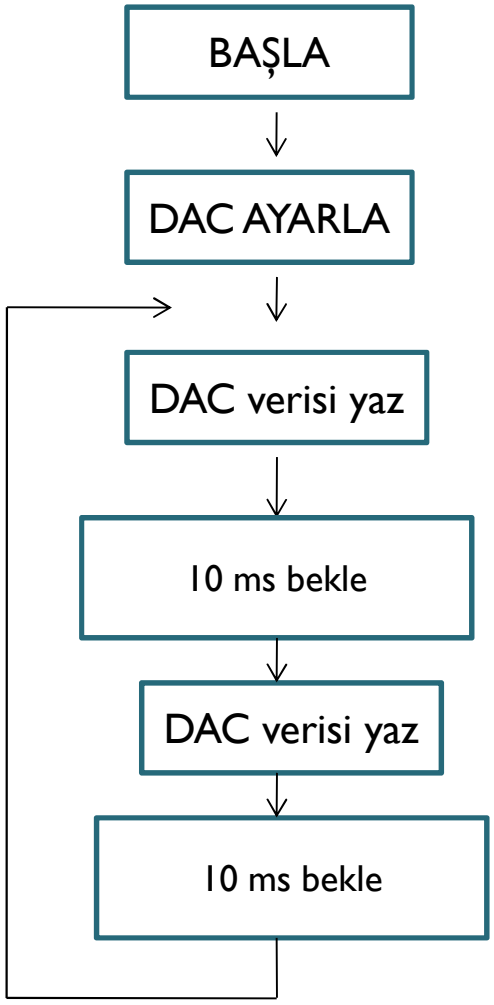
ÖRNEK: V_{DD}=3.3V için, DAC çıkışı 0-V_{DD} olmak üzere, çıkışta 2.25V elde edilmesi amacıyla DAC kontrol saklayıcılarına uygun değerleri yazınız.

12 bit değer: 4095*2.25/3.3=2792=0AE8H

```
MOV     ADCCON1, #80H
MOV     DACCON, #2DH
MOV     DAC0H,  #00AH
MOV     DAC0L,  #0E8H
```

ÖRNEK

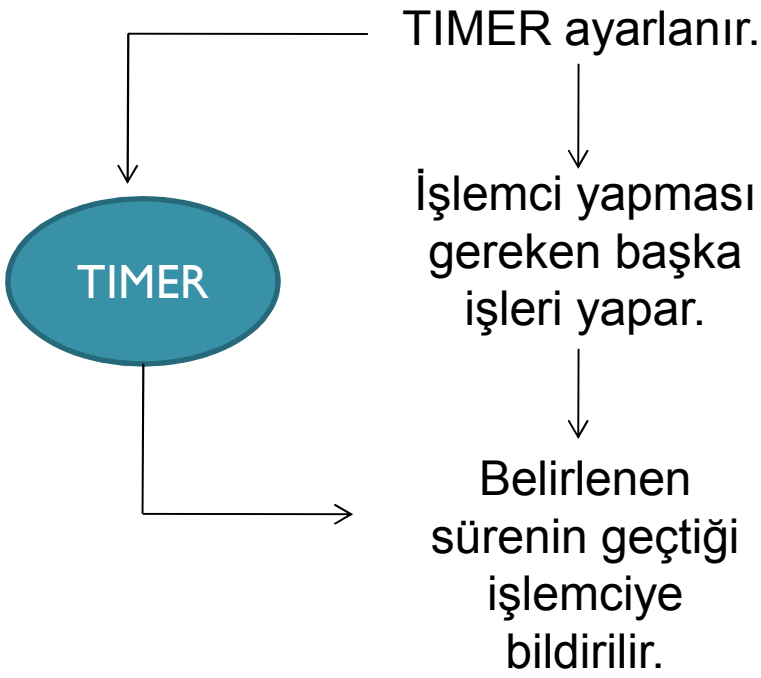
DAC0 kullanılarak çıkışta 50 Hz'lik kare dalga oluşturulmak isteniyor. Kare dalganın alt ve üst gerilim değerleri sırasıyla 1V ve 3V olacaktır. Gerekli programı yazınız.




```
$MOD841         ; use 8052 predefined symbols
CSEG
ORG 0000H       ;start code from 0000H
BASLA:
MOV         ADCCON1, #80H
MOV         DACCON, #2DH
DACYAZ:
MOV         DAC0H,  #004H
MOV         DAC0L,  #0D9H       ;1V
MOV         A, #01H
CALL        DELAY

MOV         DAC0H,  #00EH
MOV         DAC0L,  #08BH       ;3V
MOV         A, #01H
CALL        DELAY
JMP         DACYAZ
;_____________________________________
DELAY:
            MOV     R5,A
DLY2:       MOV     R7,#090h
DLY1:       MOV     R6,#0FFh
            DJNZ    R6,$
            DJNZ    R7,DLY1
            DJNZ    R5,DLY2
            RET
END
```

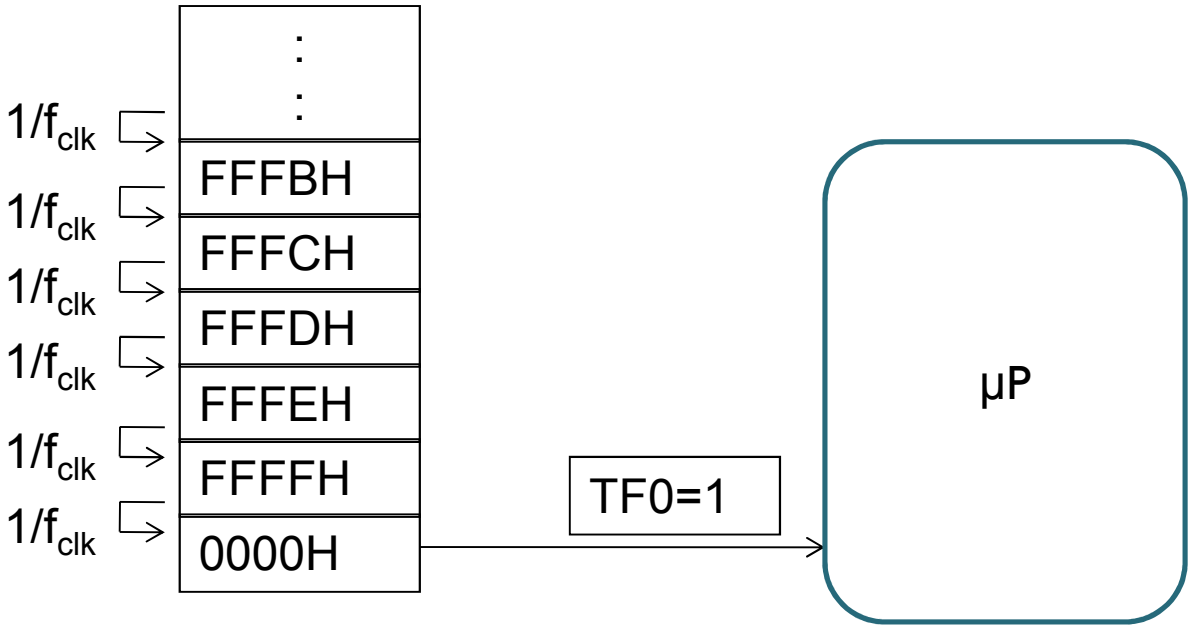
TIMER (Zamanlayıcı)

- Bir işin periyodik olarak tekrarlanması için ya da belirli bir süre beklemek amacıyla TIMER birimleri kullanılabilir.
- TIMER, işlemci tarafından belirli bir süreye ayarlanır.
- TIMER, işlemciden bağımsız olarak çalışır. Belirli bir süre geçtiğinde işlemciye bunu bildirir.
- ADC'nin periyodik olarak çalışması ve seri haberleşme (UART) biriminin çalışması için de TIMER kullanılır.



TIMER (Zamanlayıcı)

- ADUC 841 içinde 3 adet 16 bitlik TIMER bulunmaktadır. (Timer0, Timer1, Timer2)
- Her timer sayacı yüksek ve düşük anlamlı byte'a sahiptir. (THx,TLx)
- Timer kurulduktan sonra, timer sayacı ayarlandığı değerden itibaren birer birer artırılır.
- Timer sayacı FFFF değerine ulaştığı anda, TFx(Timer Interrupt) biti bir olur.
- Timer bir kez daha kullanılacaksa, Timer sayacı yeniden ayarlanmalıdır.



TMOD Timer/Counter 0 and 1 Mode Register

TMOD saklayıcısı, Timer0 ve Timer1'in çalışma modlarını düzenler.

Bit	İsim	Açıklama
7	**GATE**	Timer1 kontrolünün kim tarafından yapılacağını bildirir. 0 yapılırsa, TR1 kontrol biti Timer'ı kontrol eder. 1 yapılırsa, dış kesme hattı (INT1) tarafından kontrol edilir.
6	**C/T**	Timer1'i "Counter" ya da "Timer" olarak çalıştırır. (Counter seçildiğinde, Timer sayacı T1 pini ile artırılır) Counter=1, Timer=0
5	**M1**	M1 ve M0, Timer1 modunu belirler. M1=0, M0=1 yapılırsa, 16-bit Timer olarak çalışır.
4	**M0**	
3	**GATE**	Timer0 kontrolünün kim tarafından yapılacağını bildirir. 0 yapılırsa TR0 kontrol biti Timer'ı kontrol eder. 1 yapılırsa, dış kesme hattı (INT0) tarafından kontrol edilir.
2	**C/T**	Timer0'i "Counter" ya da "Timer" olarak çalıştırır. (Counter seçildiğinde, Timer sayacı T0 pini ile artırılır) Counter=1, Timer=0
1	**M1**	M1 ve M0, Timer0 modunu belirler. M1=0, M0=1 yapılırsa, 16-bit Timer olarak çalışır.
0	**M0**	

ÖRNEK: Timer0, 16 bit olarak, zamanlayıcı modunda çalışacaktır. Gerekli kodu yazınız.

MOV TMOD, #01H

TCON — **Timer/Counter 0 and 1 Control Register**

TCON saklayıcısı Timer 1 ve 0'ın kontrol edilmesini ve kullanılmasını sağlar.

Bit	İsim	Açıklama
7	**TF1**	Timer1 taşma biti, Timer1 sayacı FFFF adresine ulaştıktan sonra bir daha artırılırsa TF1 biti **donanım tarafından** 1 yapılır. Kesme kullanılıyorsa sıfıra çekilir, kullanılmıyorsa program içinde sıfırlanmalıdır.
6	**TR1**	Timer1 kontrol biti. 1: Timer1 aktif 0: Timer1 kapalı
5	**TF0**	Timer0 taşma biti, Timer0 sayacı FFFF adresine ulaştıktan sonra bir daha artırılırsa TF0 biti **donanım tarafından** 1 yapılır. Kesme kullanılıyorsa sıfıra çekilir, kullanılmıyorsa program içinde sıfırlanmalıdır.
4	**TR0**	Timer0 kontrol biti. 1: Timer0 aktif 0: Timer0 kapalı
3	**IE1**	Son 4 bit dış ortamdan tetiklenme sağlamak için ayarlanır. (Kullanılmadığı zaman 0,0,0,0 olarak ayarlanacaklardır)
2	**IT1**	
1	**IE0**	
0	**IT0**	

Timer/Counter 0 and 1 Data Registers

TH0 and TL0	Timer 0 sayaç saklayıcısı (16 bit)
TH1 and TL1	Timer 1 sayaç saklayıcısı (16 bit)

ÖRNEK: Timer0'ı 100us'ye ayarlayalım.

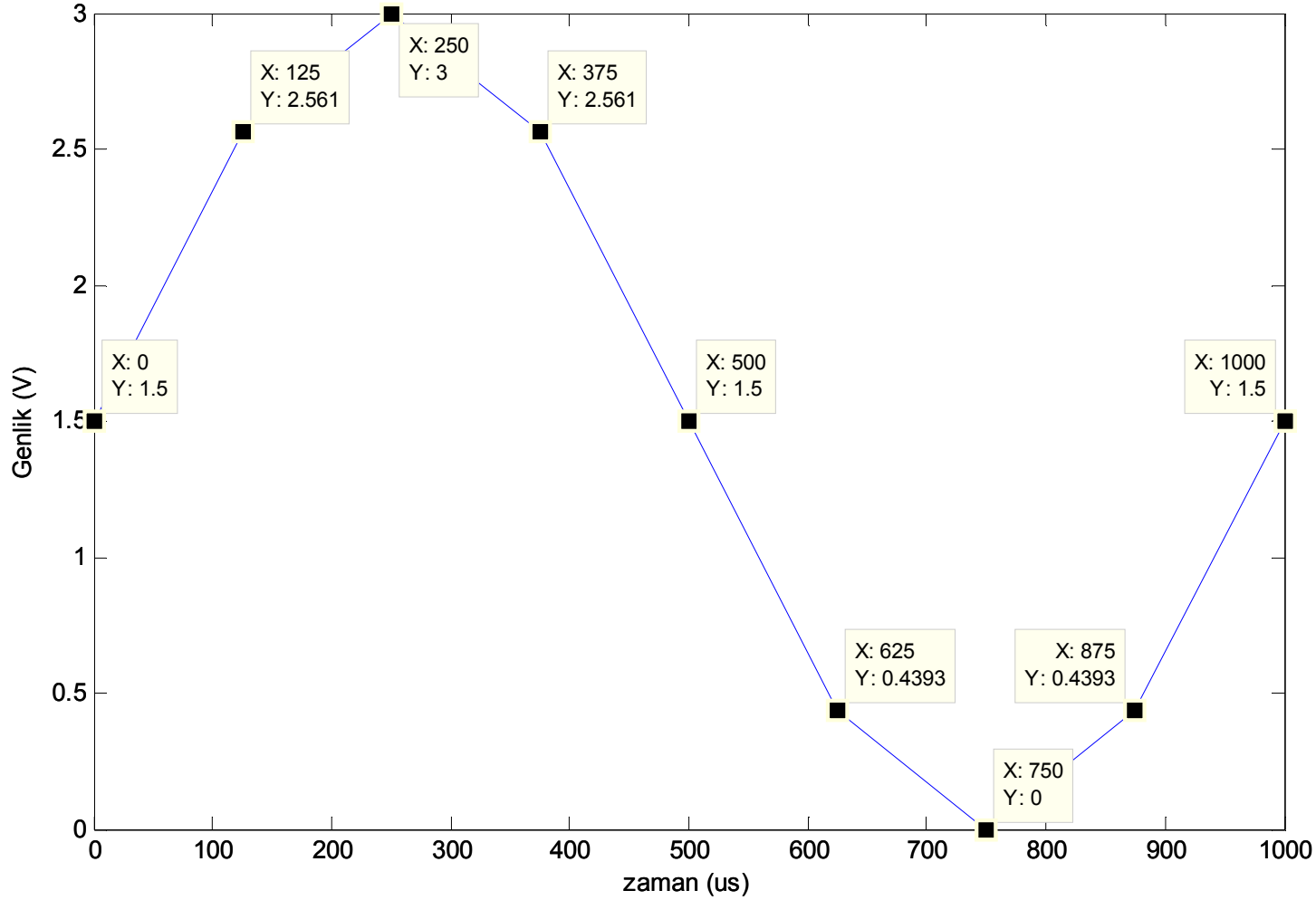
Sistem saat frekansı: 11.0592 MHz
Timer periyodu: 1/11.0592MHz=90.42ns

100 µs=90.42ns (65536-x)
X=64430= FBAE

```
MOV  TMOD, #01H
MOV  TCON, #00H
MOV  TH0, #0FBH
MOV  TL0, #0AEH
SETB TR0     ; Timer0'ı aktifleştir
```

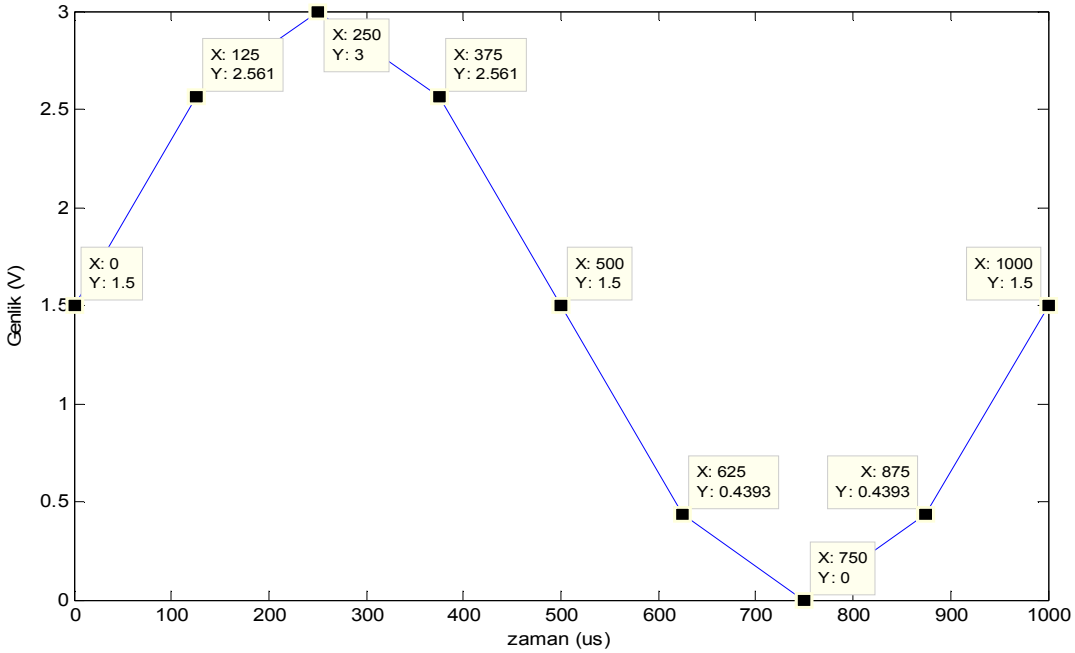
ÖRNEK: 1kHz frekansında sinüs işareti üretilmek istenmektedir. Bu amaçla program belleğinin 1000H adresinden itibaren 8kHz'de örneklenmiş bir sinüs tablosu bulunmalıdır. Sinüs işareti 0V-3V arasında olacaktır. Gerekli programı yazınız.

1. Adım: Sinüs tablo değerlerinin oluşturulması



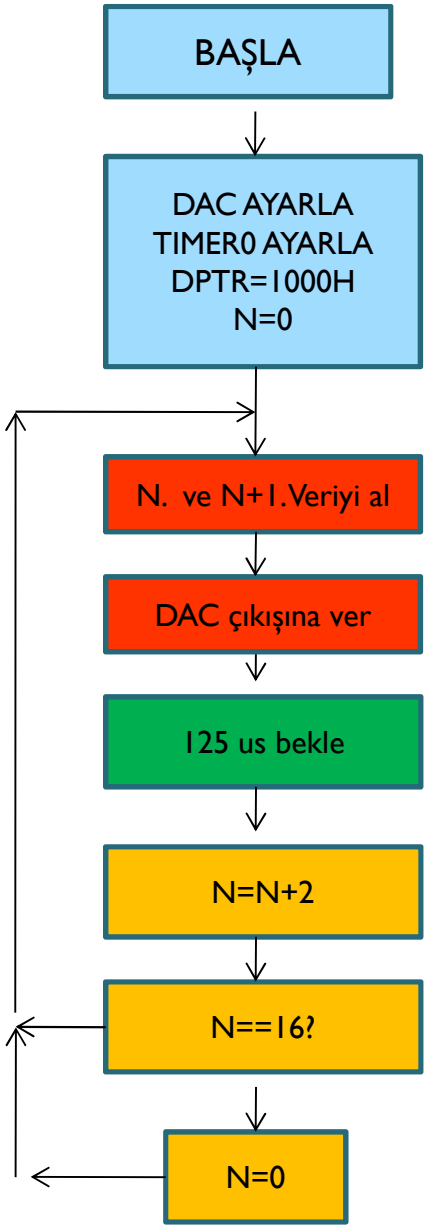
Sinüs tablosu değerleri:

no	gerilim	12 bit ondalık	12 bit 16'lık sayı
1	1.5	1861	0745H
2	2.56	3177	0C69H
3	3	3723	0E8BH
4	2.56	3177	0C69H
5	1.5	1861	0745H
6	0.4393	545	0221H
7	0	0	0000H
8	0.4393	545	0221H




```
ORG 1000H
DB      007H, 045H      ;1.5V
DB      00CH, 069H      ;2.56V
DB      00EH, 08BH      ;3V
DB      00CH, 069H      ;2.56V
DB      007H, 045H      ;1.5V
DB      002H, 021H      ;0.44V
DB      000H, 000H      ;0V
DB      002H, 021H      ;0.44V
```

2. Adım: Programın oluşturulması




```
$MOD841       ; use 8052 predefined symbols
CSEG
ORG 0000H           ;start code from 0000H
BASLA:
MOV             ADCCON1, #80H
MOV             DACCON, #2DH

MOV  TMOD, #01H         ;Timer0 16 bit
MOV  TCON, #00H
MOV  DPTR, SINTABLO     ;DPTR = 1000H
MOV  R0,#00H            ;N=0

DACYAZ:
MOV             A,R0
MOVC            A,@A+DPTR   ;Tablo degeri al (H)
MOV             R1,A

INC             R0          ;N=N+1
MOV             A,R0
MOVC            A,@A+DPTR   ;Tablo degeri al (L)
MOV             R2,A

MOV             DAC0H, R1   ;DAC çıkışına yaz (H)
MOV             DAC0L, R2   ;DAC çıkışına yaz (L)

CLR             TF0
MOV             TH0, #0FAH
MOV             TL0, #09AH
SETB            TR0
JNB             TF0, $

INC             R0           ;N=N+2
ANL             00H, #0FH
JMP             DACYAZ

ORG 1000H
SINTABLO:
DB              007H, 045H      ;1.5V
DB              00CH, 069H      ;2.56V
DB              00EH, 08BH      ;3V
DB              00CH, 069H      ;2.56V
DB              007H, 045H      ;1.5V
DB              002H, 021H      ;0.44V
DB              000H, 000H      ;0V
DB              002H, 021H      ;0.44V

END
```

UART (Evrensel Asenkron Alıcı/Verici)

- UART, donanım elemanları tarafından sıkça kullanılan bir haberleşme protokolüdür.

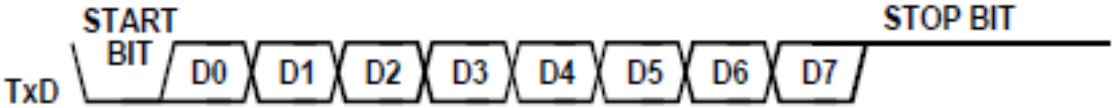
- Veri tek hat üzerinden karşı tarafa iletilir (**Tx**),
  tek hat üzerinden de alınır (**Rx**).

- ADuC 841 denetleyicisinde, UART haberleşmesi için, bir Tx ve bir Rx hattı bulunmaktadır.

- Seri haberleşme protokolü SFR'ler aracılığıyla kurulup veri aktarımı kolayca yapılabilir.

- UART çeşitli aktarım hızlarına sahiptir. Bu hızlar, TIMER3 ile ayarlanır.

UART verisi:



UART kullanımı

SCON UART Serial Port Control Register

SCON saklayıcısı UART konfigürasyonunun yapılmasını sağlar.

Bit	İsim	Açıklama
7	**SM0**	SM0 ve SM1 seri haberleşme mod'unu seçer. Bu mod'lar aşağıda belirtilmiştir: •Mod 0 (SM0=0, SM1=0): Kaydırma saklayıcısı türü. Bit gönderim hızı sabittir. **•Mod 1 (SM0=0, SM1=1): 8 bit UART haberleşmesi. Bit gönderim hızı seçilebilir. Bir başla biti ile bir dur biti karaktere eklenir. Genellikle bu mod kullanılır.** •Mod 2 (SM0=1, SM1=0): 9 bit UART haberleşmesi. Sabit bit gönderim hızı. •Mod 3(SM0=1 SM1=1): 9 bit UART haberleşmesi, seçilebilir bit gönderim hızı.
6	**SM1**	
5	**SM2**	SM2 biti çoklu işlemci haberleşmesinin olması durumunda 1 yapılır.
4	**REN**	REN biti, seri haberleşmeden veri alımının aktif olduğunu belirtir.
3	**TB8**	TB8 ve RB8 Mod 2 ve 3 için 9. bitler olmaktadırlar.
2	**RB8**	
1	**TI**	Bir karakter gönderildiğinde , TI kesme biti donanım tarafından 1 yapılır. Daha sonra kullanıcı tarafından 0 yapılmalıdır.
0	**RI**	Bir karakter alındığında, RI kesme biti donanım tarafından 1 yapılır. Daha sonra kullanıcı tarafından 0 yapılmalıdır.

ÖRNEK: MOD1, Receive Enable için SCON biti: 52H olarak ayarlanır.

```
MOV     SCON,#52H ;          0 1 0 1   0 0 1 0
```

UART kullanımı

Veri aktarım hızının (Baud Rate) ayarlanması

- Seri haberleşmede veri bit bit aktarılır.
- Doğru bir biçimde haberleşmek için iki bit arasında belirli bir süre geçmesi gereklidir. Bu süre “baud rate” ile ilişkilidir.
- Haberleşmenin sağlanması için Baud Rate değeri hem alıcı hem de verici tarafından bilinmelidir.
- Veri aktarım hızını ayarlamak için Timer3, T3CON ve T3FD kontrol saklayıcıları ile ayarlanmalıdır.

T3CON: 1 0 0 0 0 DIV2 DIV1 DIV0

DIV değeri

$$DIV = \frac{\log\left(\frac{f_{CORE}}{16 \times Baud\ Rate}\right)}{\log(2)}$$

$$T3FD = \frac{2 \times f_{CORE}}{2^{DIV-1} \times Baud\ Rate} - 64$$

ÖRNEK: 11.0592 MHz'lik sistem saatine sahip ADuC 841 ile 38400 Baud Rate hızında seri iletişim yapılmak isteniyor. Gerekli saklayıcıları ayarlayınız.

$$DIV = \frac{\log(\frac{f_{CORE}}{16 \cdot BaudRate})}{\log(2)} = \frac{\log(\frac{11059200}{16 \times 38400})}{\log(2)} = \frac{\log(18)}{\log(2)} = 4.17 \approx 4$$

! DIV değeri için, bulunan sonucun tamsayı kısmı alınmalıdır.

T3CON: 1 0 0 0 0 1 0 0 = 84H

DIV değeri

$$T3FD = \frac{2 \cdot f_{CORE}}{2^{DIV-1} \cdot BaudRate} - 64 = \frac{2 \cdot 11059200}{2^3 \cdot 38400} - 64 = 72 - 64 = 8$$

```
MOV     SCON,   #52H
MOV     T3CON,  #84H
MOV     T3FD,   #08H
```

UART kullanımı

Veri gönderimi ve alımı

- Veri gönderimi ve alımı için **SBUF** saklayıcısı kullanılır. Bu saklayıcıya veri yazmak, verinin seri porttan iletilmesi, saklayıcıyı okumak, dışarıdan gelen veriyi okumak anlamına gelir.

Veri gönderimi

- Öncelikle, daha önceki veri gönderme işleminin tamamlanmış olduğunu sorgulamak için TI bitine bakılır. Bu bit 1 ise seri port gönderime hazır demektir.

```
JNB       TI,       $
CLR       TI
MOV       SBUF, A
```

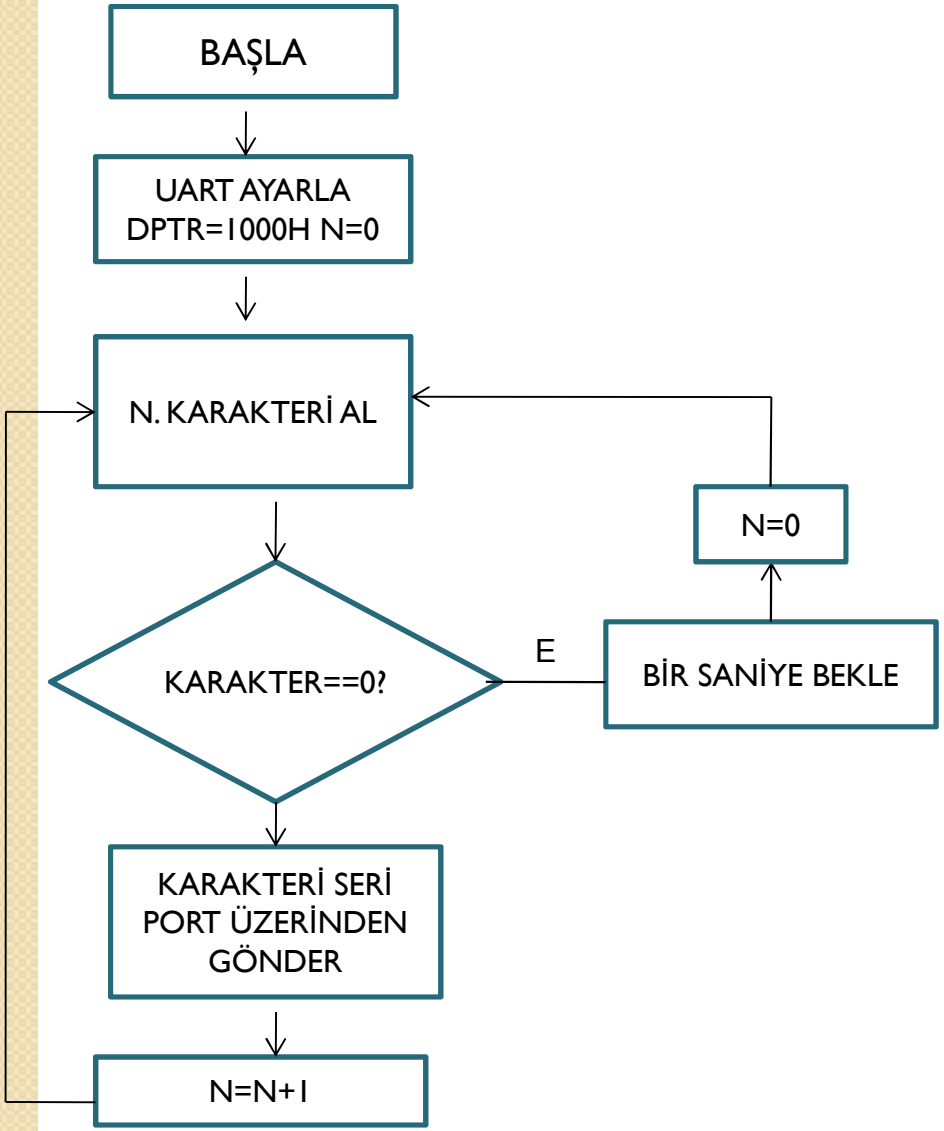
Veri alımı

- Öncelikle, yeni veri gelip gelmediği öğrenilmelidir. Bu amaçla RI bitine bakılır. 1 ise yeni veri gelmiştir, SBUF saklayıcısından alınabilir.

```
JNB       RI,       $
CLR       RI
MOV       A, SBUF
```

ÖRNEK: Ekrana belirli bir yazı yazma

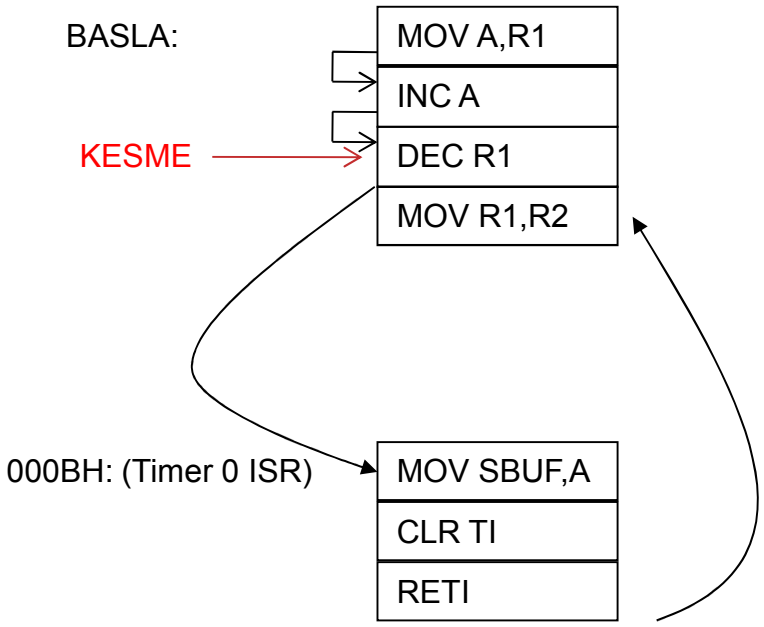
Program belleğinde 1000H adresinde bulunan karakter dizisi, saniyede bir seri port üzerinden bilgisayara aktarılacaktır. UART 38400 hızında çalışacaktır. Kodu yazınız.




```
        MOV     T3CON,#084H
        MOV     T3FD, #008H
        MOV     SCON, #052H

BASLA:  MOV     DPTR,#YAZI
        MOV     B,#0FFH

GOND:   INC     B
        MOV     A,B
        MOVC    A,@A+DPTR
        JZ      SON

        JNB     TI,$
        CLR     TI
        MOV     SBUF,A
        JMP     GOND
SON:
        MOV     A,#064H
        CALL    DELAY
        JMP     BASLA

ORG     1000H
YAZI:
DB      10,13
DB      'MERHABA ADUC 841 '
DB      00H
```

KESMELER (Interrupts)

- Kesme, normal program çalışmasını kesen bir mekanizmayı tanımlar.
- Programların tek başına, dış ortamdan gelen donanım isteklerini karşılamaları için, belirli saklayıcıları sürekli kontrol etmeleri gerekir.
- Kesme kullanıldığında, programların donanımları sürekli kontrol etmesine gerek kalmaz, kesme geldiğinde program kendiliğinden gerekli alt programa dallanır.
- Kesme için tanımlı alt programların adresleri belirlidir. Kullanıcı bu adreslere gerekli kodu yazarak, kesme servis rutini (ISR) denilen alt programlar oluşturur.



KESMELER (Interrupts)

KESME KONTROL SAKLAYILARI

IE — Interrupt Enable Register

Kesmelerin hangilerinin dikkate alınacağı, hangilerinin alınmayacağını belirler.

Bit No.	Name	Description
7	EA	Set by the user to Enable, or cleared to Disable all interrupt sources.
6	EADC	Set by the user to Enable, or cleared to Disable ADC interrupts.
5	ET2	Set by the user to Enable, or cleared to Disable Timer 2 interrupts.
4	ES	Set by the user to Enable, or cleared to Disable UART serial port interrupts.
3	ET1	Set by the user to Enable, or cleared to Disable 0 Timer 1 interrupts.
2	EX1	Set by the user to Enable, or cleared to Disable External Interrupt 1.
1	ET0	Set by the user to Enable, or cleared to Disable Timer 0 interrupts.
0	EX0	Set by the user to Enable, or cleared to Disable external interrupt 0 .

ÖRNEK: Yalnızca Timer0 kesmesi kullanılacaksa, IE saklayıcısı aşağıdaki gibi ayarlanır.

```
MOV       IE, #82H ;        1 0 0 0 0 0 1 0
```

Ya da,

```
          SETB      ET0;
          SETB      EA ;
```

KESMELER (Interrupts)

KESME VEKTÖR TABLOSU

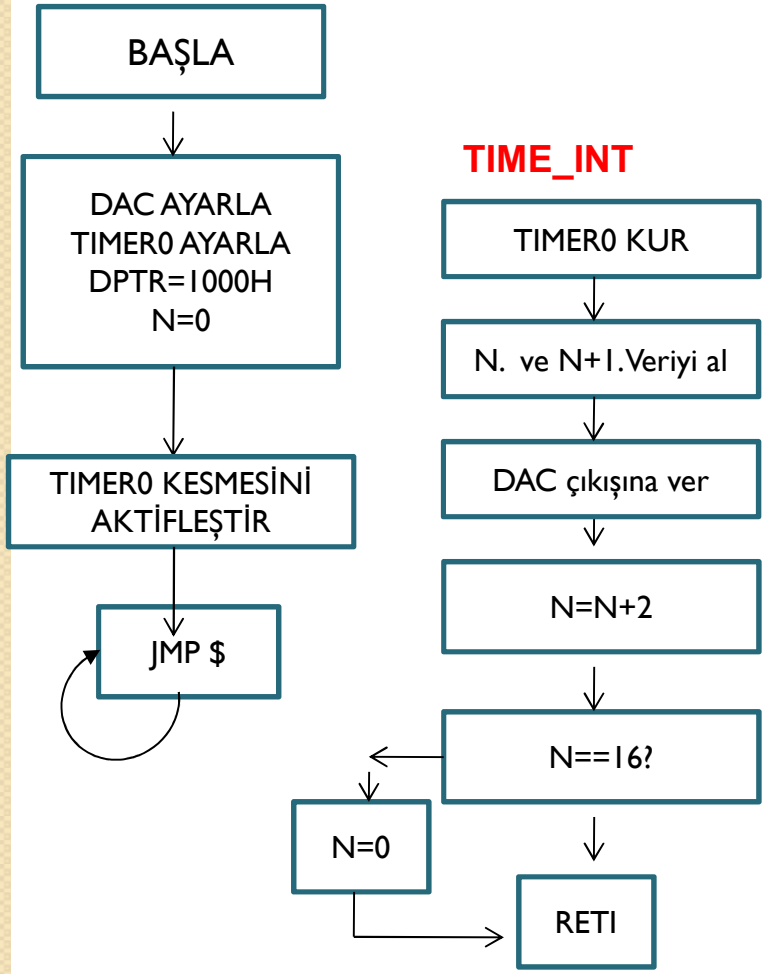
Bir kesme geldiğinde (eğer aktifleştirilmişse), program sayacı (PC) kesmenin türüne göre aşağıdaki adreslerden birine dallanır. Bu adreslerde genellikle, ISR denilen kesme servis rutini alt programlarına yönlendirmeler bulunur.

Table 39. Interrupt Vector Addresses

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
ADCI	0033H
ISPI/I2CI	003BH
PSMI	0043H
TII	0053H
WDS	005BH

```
ORG   0000H
JMP         BASLA

ORG   000BH
JMP         TIME_INT

ORG         0033H
JMP         ADC_KESME
BASLA:
            .
            .
            .


TIME_INT:
            .
            .
            RETI

ADC_KESME:
            .
            .
            RETI
```

ÖRNEK: DAC ile sinüs işareti üretimi örneğindeki zamanlayıcıyı kesme kullanarak çalıştırınız.




```
ORG  0000H
JMP          BASLA

ORG          000BH
JMP          TIME_INT

BASLA:
             MOV          ADCCON1, #80H
             MOV          DACCON, #2DH

             MOV  TMOD,   #01H
             MOV  TCON,   #00H
             MOV  TH0,    #0FAH
             MOV  TL0,    #09AH
             MOV  DPTR,   #SINTABLO
             MOV  R0,     #00H

             SETB         TR0
             SETB         ET0
             SETB         EA
             JMP          $

TIME_INT:

             MOV          A,R0
             MOVC         A,@A+DPTR
             MOV          R1,A
             INC          R0
             MOV          A,R0
             MOVC         A,@A+DPTR
             MOV          R2,A
             MOV          DAC0H, R1
             MOV          DAC0L, R2
             INC          R0
             ANL          00H, #0FH
             MOV          TH0, #0FAH
             MOV          TL0, #09AH
             SETB         TR0
             RETI
ORG 1000H
SINTABLO:
DB           007H, 045H   ;1.5V
DB           00CH, 069H   ;2.56V
DB           00EH, 08BH   ;3V
DB           00CH, 069H   ;2.56V
DB           007H, 045H   ;1.5V
DB           002H, 021H   ;0.44V
DB           000H, 000H   ;0V
DB           002H, 021H   ;0.44V

END
```

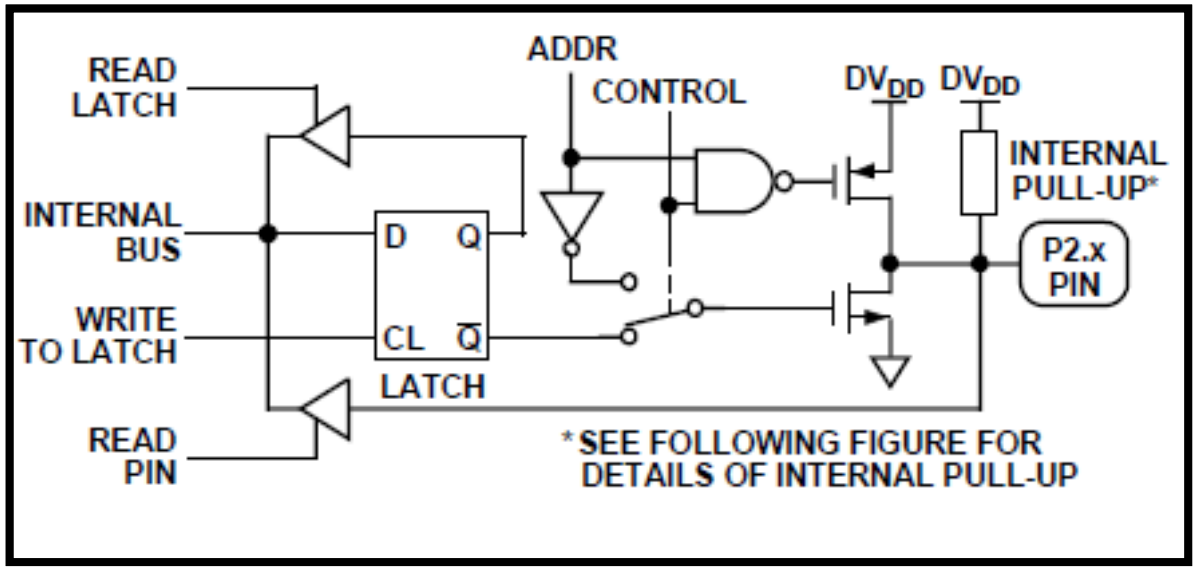
PORT2 ve Kontrolü

PORT2, çift yönlü bir giriş çıkış portudur.

P2.0-P2.7 uçlarına bit olarak ulaşılabilir.

P2.x bitine bir değer yazılması halinde bu çıkışta görülebilir. Çıkıştan fazla akım çekmemek için genellikle uçlar buffer ile korunur.

P2.x giriş olarak kullanılacaksa, çıkıştan toprağa çekilir. Normal durumda bu pin 1 (Vdd) değerindedir.



Örnek:

P2.7 ucundaki LED'i yakıp söndürme:

```
$MOD841                 ; use 8052 predefined symbols
LED    EQU    P2.7      ; P2.7 is red LED on eval board
;_____________________________________________________________________
                        ; MAIN PROGRAM
CSEG

ORG  0000h
             MOV        A,#001      ; set delay length
BLINK:       CPL        LED         ; flash (complement) the red LED
             CALL       DELAY       ; call software delay
             JMP        BLINK       ; repeat indefinitely

;_____________________________________________________________________
                        ; SUBROUTINES
DELAY:       ; delay 10ms X A
             MOV        R5,A
DLY2:        MOV        R7,#048h
DLY1:        MOV        R6,#0FFh
             DJNZ       R6,$
             DJNZ       R7,DLY1
             DJNZ       R5,DLY2
             RET_____________________________________________________________

END
```

Örnek:

P2.6 ucundaki değerin kontrol edilmesi:

```
$MOD841                 ; use 8052 predefined symbols
LED     EQU     P2.7    ; P2.7 is red LED on eval board
BUTTON  EQU     P2.6
;_____________________________________________________________________
                        ; MAIN PROGRAM
CSEG

ORG  0000h
        MOV     A,#010      ; set delay length

BLINK:  JB      BUTTON, $
        CPL     LED         ; flash (complement) the red LED
        CALL    DELAY       ; call software delay
        JMP     BLINK       ; repeat indefinitely

;_____________________________________________________________________
                        ; SUBROUTINES
DELAY:  ; delay 10ms X A
        MOV     R5,A
DLY2:   MOV     R7,#048h
DLY1:   MOV     R6,#0FFh
        DJNZ    R6,$
        DJNZ    R7,DLY1
        DJNZ    R5,DLY2
        RET_____________________________________________________________

END
```

Kaynak

- www.analog.com - ADuC 841 data sheet

Dinlediğiniz için Teşekkürler