Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

9. ADUC814 MİNİKİT

9.1 MİNİKİT DONANIM

ADuC814 Minikit ADuC814 chipine ek olarak birkaç direnç ve bir kapasitörden başka bir özellik getirmemesine rağmen bireysel kullanıcılar için kullanması çok zor SSMD kılıflı yapıyı sıradan DIP bir entegre gibi kullanılabilmesine olanak sağlar.

Ancak chip donanımının düzgün çalışabilmesi ve bilgisayarınızın COM portu ile haberleşebilmesi için bazı ek donanım ürünlerine ihtiyaç duymaktadır. Bunlar:

32,768 kHz Clock Kristal
1K ohm Direnç
Push Button
4 dişli Pin Header
RS-232 Ara Bağlantı Kablosu (Kitle Verilir)

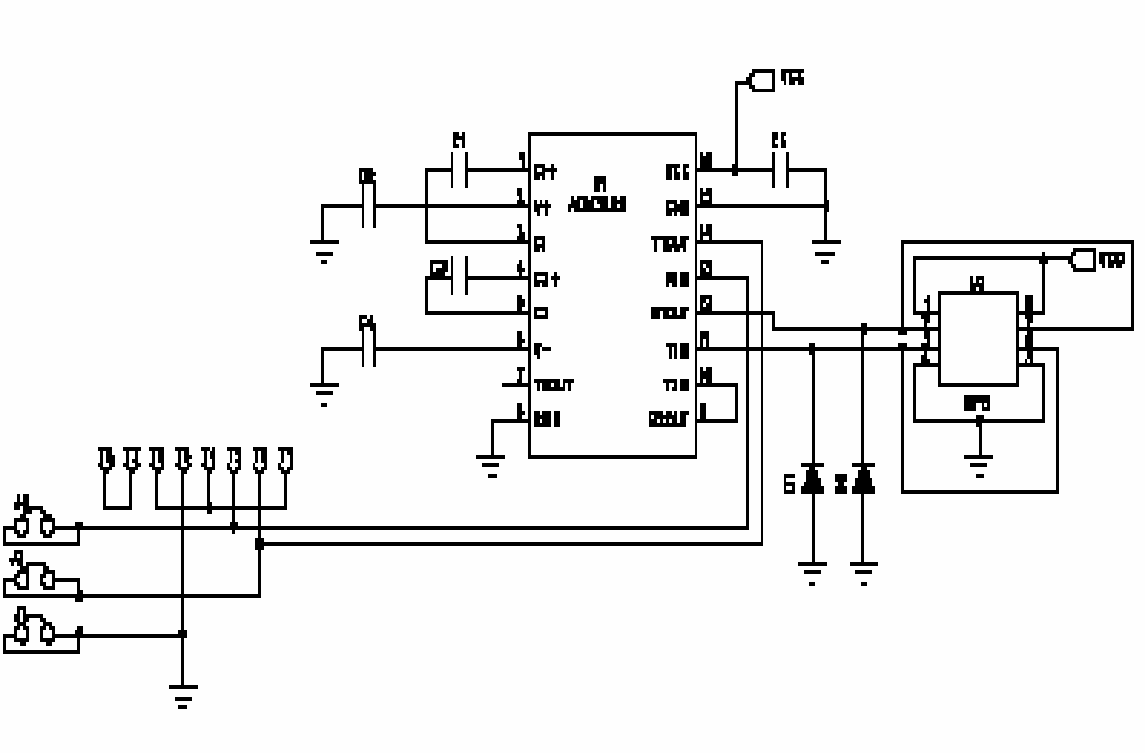
Bu ürünlerden bazıları kit ile verilirken bazılarının ise ayrıca tedarik edilmesi gerekmektedir.Kitimizin paketinde MiniKit Dip Board ve RS-232 Ara Bağlantı Kablosu bulunmaktadır.

RS-232 Kablosu üzerinde ADM3202 kodlu Analog Devices RS-232 Entegresini bulundurur. Bu entegre beslemesini bilgisayardan değil ADuC814MK'nın besleme kaynağından alır. Chipten gelen RxD ve TxD pin bağlantılarının ters olmamasına çok dikkat edilmelidir.

RS-232 Ara Bağlantı Kablosu

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

RS232 Ara Bağlantı Kablosu üzerinde yer alan RS2327/TTL level converter (+/- 12V ... 0/5 V seviye dönüştürücü) devresine ait devre şeması aşagıdaki şekilde görülmektedir.



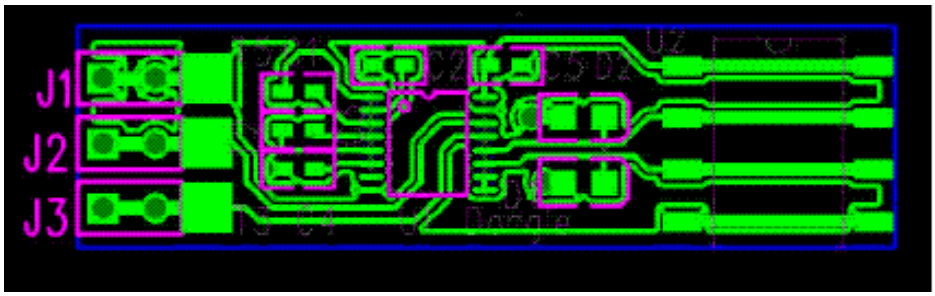
RS232 Ara Bağlantı Kablosu üzerinde yer alan RS232 / TTL level converter (+/- 12V ... 0/5 V seviye dönüştürücü) devresine ait malzeme listesi aşagıdaki tabloda yer almaktadır.

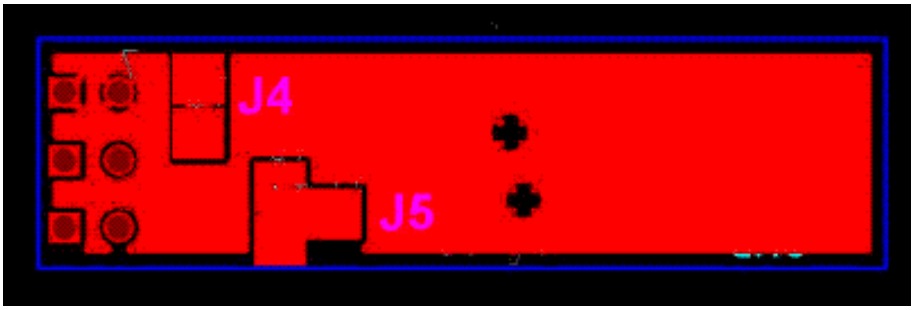
Qty	Ref Des	Description	Supplier	Part Number
1	U1	ADM3202ARU	ADI (Free Issue)	
1	U2	4 Pin Single Row Horizontal SMD Header	Samtec	SSM-104-L-SH
5	C1, C2, C3, C4, C5	0.1uF SMD Multilayer Ceramic Cap, 0603 Case	Farnell	317-287
2	D1, D2	5.6V Zener Diodes SOT23 Package	Farnell	354-6901

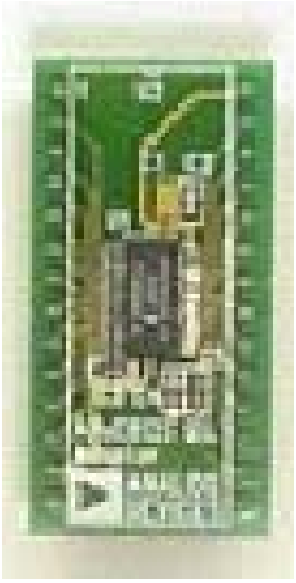
D TYPE Connector	PCB Board
Pin 1	J5
Pin 2	J2
Pin 3	J3
Pin 4	J5
Pin 5	J1
Pin 6	J5
Pin 7	J4
Pin 8	J4
Pin 9	NC

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

RS232 Ara Bağlantı Kablosu üzerinde yer alan RS2327/TTL level converter (+/- 12V ... 0/5 V seviye dönüştürücü) devresine ait PCB gösterimi aşağıda görülmektedir.



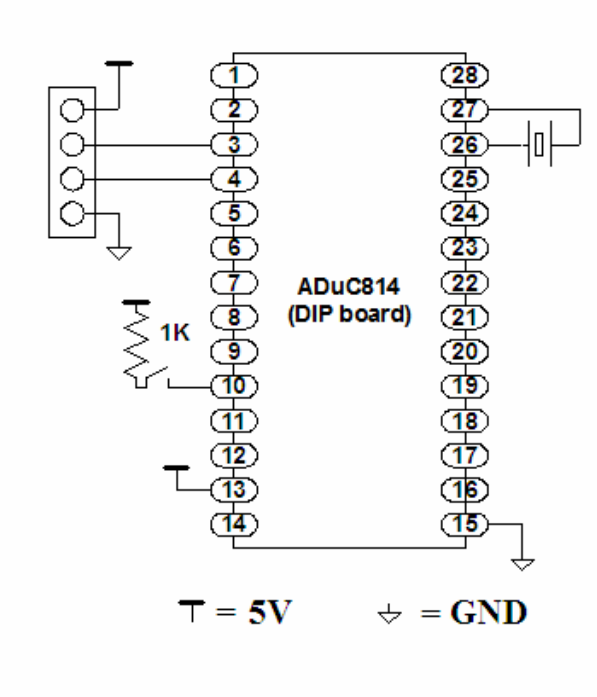




ADuC814 Minikit herhangi standart 28 bacaklı entegre ile çok yakın boy ve genişliktedir. Ayrıca bu tip entegrelere göre üretilmiş entegre soketlerine uyar.

Minikitin Standart Bağlantı Şeması ise aşağıdaki şekilde gösterilmiştir:

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr



MiniKit Standart Bağlantı Şeması

9.2 QUICK START KİTİ İLE SAĞLANAN YARDIMCI YAZILIMLAR

Bu bölümde ADuC814 ve diğer Analog Devices MCV ürünleri ile kullanabileceğiniz çok faydalı yardımcı programların ADuC814 Minikit kullanımına dair püf noktalarını inceleyeceğiz. Programların detaylı açıklamalarını 10.2 ADuC814EB Eval Board Getting Started Guide bölümünde inceleyebilirsiniz.

9.2.1 ASM51 (Metalink 8051 Cross Assembler)

Intelin standart 8051 assembleri baz alınarak düzenlenmiş bu yazılımın tek farkı compile edilen *.asm dosyasından *.hex ve *.lst dosyaları oluşturmasıdır.

Bu programın kullanımına dair en önemli püf noktası assembler edeceğiniz dosya ile aynı klasörde bu programı çalıştırmak olacaktır tabi 814 tabanlı programlarda kullanılacak olan MOD814 dosyasının da ASM51.EXE ile aynı klasörde olmasıdır.Windows bütün versiyonları için komut sistemini açarak programınız,ASM51.EXE ve MOD814 dosyanızın bulunduğu klasörü çağırarak

..\ASM51 <dosya adı>.asm

komutuyla dosyanızı compile edebilir ve programınızdaki hata sayısını öğrenebilirsiniz.Bu tür bi yaklaşımın faydası ASM51.EXE dosyasını Windows gezgininden çalıştırdığınızda programın işlemini yürütüp siz hata sayınızı göremeden pencerenin kapanmasını engellemesidir.

9.2.2 WSD (Windows Serial Downloader)

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Bu uygulama windows işletim sisteminizi kullanan basit bir makine kodu yükleme programıdır.İlgili bölümde anlatılan standart kullanım tarzına ek olarak unutulmaması gereken bazı ek kullanım standartları vardır.

Öncelikle bilgisayarınızın seri port ayarları her ne kadar WSD standartlarına uyuyorsa da yine de Configuration penceresinde uygun COM portunun ve Watch Crystal seçili olduğundan emin olun.

Kiti download için uygun koşullara getirip. Besleme bağlantılarını ve seri konnektörün bağlantısını kontrol ettikten sonra, WSD programında Reset tuşuna basın ilk basışta Fail uyarısı vermesi muhtemeldir.Donanımınızı tekrar kontrol etmeden önce 2-3 kez daha Reset komutunu uygulayın.Fail durumu devam ediyorsa bağlantılarınızı kontrol edin ve başka bir seri port tabanlı programın çalışmadığından emin olun.

9.2.3 ASPIRE IDE (Integrated Development Environment)

Aspire adından da anlaşılabileceği gibi çeşitli diğer yardımcı programların birleşmesinden oluş bir IDE programıdır. Programı kullanarak diğer bütün yazılımlarla yapabildiğimiz işlemlere ek olarak. Simulasyon işlemini gerçekleştirebiliriz.

Getting Started Guide bölümünde anlatılmayan Simulasyon işlemi için Aspire içerisindeki RUN üst menüsünden Load DeBugger seçeneklerine girerek açılan pencerede Simulator seçeneğini seçmeniz gerekir.Sıradaki işlem ise tekrar RUN menüsünden ADuC Hardware Simulator Setup seçeneğine girerek hangi MCV ‘yi simule etmek istediğinizi seçmelisiniz.Listede ADuC814 MCV ‘sinin olmaması gibi bir durum söz konusu değildir.

Her DeBugger programı kit üzerinden seri haberleşme sistemini kullanmaktadır uygulamalarda da anlatıldığı üzere Timer0 kullanan bir programı DeBug işlemine tabi tuttuğunuzda TMOD registerı ayarlanırken Timer1 ’in kapatılmamasına yada ayarlarının değiştirilmemesine dikkat ediniz. Bu işlemin alternatif yöntemi Bölüm 9.3 ‘deki ilgili uygulamada anlatılmıştır.

9.2.4 DeBugger

Aspire IDE ’sine benzer bir uygulama olan DeBugger adından da anlaşıldığı gibi On-Chip debug işlemleri için kullanılır. Programınızın *.lst uzantılı dosyalarını kullanarak işlem yapan program *.asm dosyalarını compile etmek, program simule etmek gibi işlemleri gerçekleştiremez. Ancak bu sebeple Aspire IDE ’sine göre daha seri DeBug işlemi gerçekleştirir.

Bu uygulamada da Aspire da olduğu gibi DeBug işlemi seri port üzerinden yapıldığından timer kullanılan uygulamalarda dikkat edilmelidir.

9.3 ADUC814MK İLE İLK PROJELER

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

9.3.1 BİR PORT PİNİNE BAĞLI OLAN BİR LEDİN "DELAY" ALT-RUTİNİ KULLANILARAK FLAŞÖR HALİNE GETİRİLMESİ

Önbilgi:ADuC 814 Mikrokonverter A/D ve D/A dönüşümü özelliklerinin yanısıra 8051 tabanlı bir mikrokontrolördür.Port pinlerinin yan özelliklerinin elverdiği ölçüde bazı pinler her hangi bir I/O pini olarakta kullanılabilirler.Buna rağmen her pin her hangi bir cihazı veya devre elemanını sürebilecek akımı dışarı doğru akıtamaz.ADuC 814 takılmış olan minikitimizin bazı port pinleri kolaylıkla yeterli şiddette akım sürebilirler. Bu pinler:

P1.0
P1.1
P3.2
P3.3
P3.4
P3.5
P3.6
P3.7

gibi port pinleridir. Bu pinlere kodumuzun içerisinde;

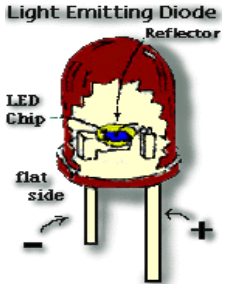
SETB Px.x

komutunu uygularsak; aksi yönde bir komut veya donanımsal bağlantı koşullanmadığı sürece Minikitmizin bu bacağından yeterli akım sürülebilir.Örneğin Anot ucunu 330Ωluk bir dirençle bu port pinine Katot ucunu ise toprağa bağladığımız bir LED bir sorun çıkarmadan ışık verebilir.

ÖnBilgi:

LED

Led (Light Emitting Diode – Işık Saçan Diyot), ışık üretimi için kullanılan, üzerinden akım geçtiğinde ışık yayan bir diyot çeşididir.Gereksiz yere ısı üretmez ve böylece çok uzun ömre sahiptir.LED 'ler küçük olmaları, düşük voltaj ile çalışmaları ve düşük güç gerektirmeleri, istenilen renk ve boyutta bulunabilmeleri ve uzun ömürlü olmaları nedeniyle kullanılması çok avantajlı ürünlerdir.
Şekil 9.1 LED ' e ait genel yapı diyagramıdır.



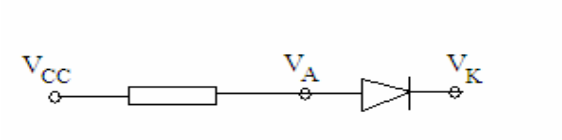
Şekil 9.2 Şekil 9.1

Şekil 9.2 ise LED 'in şematik gösterimini içerir.
Şematik gösterimde üçgen olan kısım anot; düz olan kısım ise katottur.

V_A(Lojik)	V_K (Lojik)	I_D	
0	0	0	; Led üzerinden bir akım akmaz.
0	1(5V)	0	; Led üzerinden bir akım akmaz.
1(5V)	0	10mA	; Ancak direk 5V uygulandığında Led tahrip olur,bu yüzden önüne 5V gerilimin

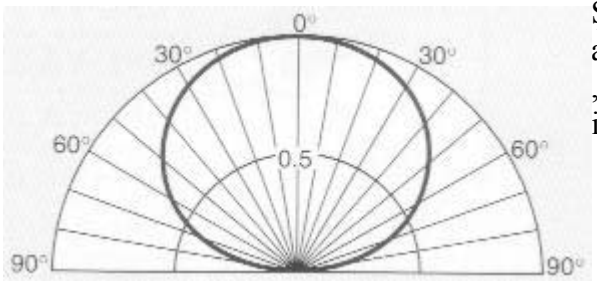
uygun gerilime düşürmek için bir direnç kullanmak gerekir.*

1(5V) 1(5V) 0 ; Led üzerinden bir akım akmaz.



* V_{CC}=5V
$V_{AK}=V_A-V_K$
V_{AK}=1.2V- 1.6V

LED ‘den yayılan ışın düz bir çizgi boyunca ilerlemez.yayılan ışının dağılımına ait bir örnek Şekil 9.3 de yer almaktadır. Şekil yeşil bir LED‘e ait ışınım diyagramıdır. Şekilden de görüldüğü gibi , en parlak görünüm karşı noktadadır, yatay eksen ile yapılan açı artıkça parlaklık azalmaktadır.

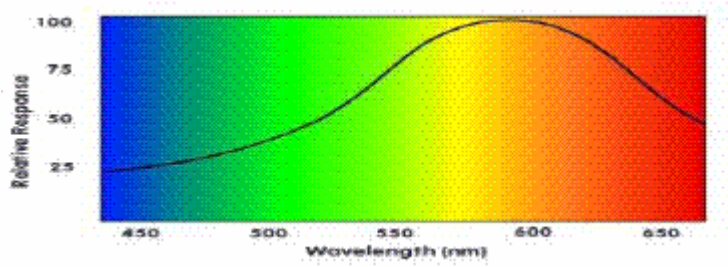


Şekil 9.3

Farklı şekillerde, renklerde ve parlaklıkta pek çok LED çeşidi vardır. LED seçiminde dikkat edeceğimiz birkaç önemli özelliği şöyle sırlayabiliriz.

- Boyut
- Şekil
- Renk
- Parlaklık
- Yayılım Şekli

LED ‘den yayılan ışığın rengi LED ‘in dalga boyu (wavelenghts) ile belilenir. Dalga boyu değiştikçe LED ‘in yaydığı ışık rengi de değişir. Şekil 9.4 dalga boyunun değişimine göre oluşan renkleri göstermektedir.

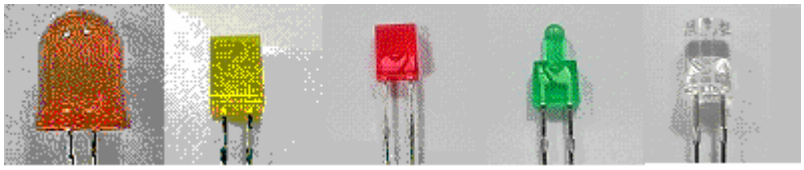


Şekil 9.4

Ayrıca LED ‘den yayılan ışık şiddeti LED milicandelle değeri ile belirlenir. Farklı milicandella değerlerine sahip LED ‘lerden yayılan ışık şiddeti de değişmektedir. Farklı kullanım amaçları

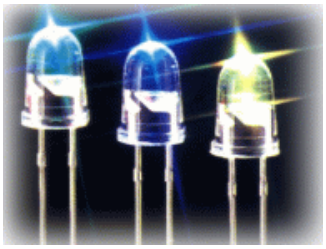
için üretilen LED 'leri şu şekilde gruplandırabiliriz : LED LAMP, SMD LED, BASE LED, LED BACKLIGHT , LED DISPLAY

LED LAMP 'lar da Standart, Süper/Ultra Parlak , Çok Renkli ,Süper Flux, Axiol ,Plastik kaplı Axiol ve IR/PT/PO olarak kendi aralarında gruplanır.
SMD LED 'ler Top LED ve (PCB) Chip LED olmak kedi arasında gruplara ayrılır. Bu gruplarda kendi içlerinde alt gruplara ayrılır.
BASE LED ve LED BACKLIGHT ' da kendi içlerinde alt gruplardan oluşurlar.
LED DISPLAY 'da arklı boyut ve digitlerde alt gruplardan oluşur.



Şekil 9.5

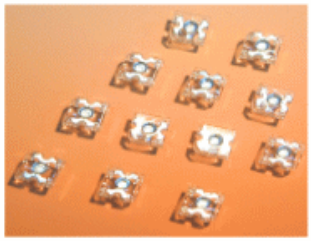
Şekil 9.5 çeşitli şekil ve renklerde Standart LED Lamp 'ları içermektedir.



Şekil 9.6

Şekil 9.7

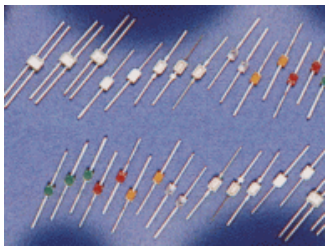
Süper/Ultra Parlak LED Lamp 'lar Şekil 9.6 ' da yer almaktadır.
Şekil 9.7 ' de çok renkli LED 'ler gösterilmiştir. Bu LED 'lerde diğerlerinden farklı olarak üç bacak yer almaktadır. Bunlardan biri birinci renk, diğeri ise ikinci renk içindir. Ortadaki bacak ise toprağa bağlanır.



Şekil 9.8

Şekil 9.9

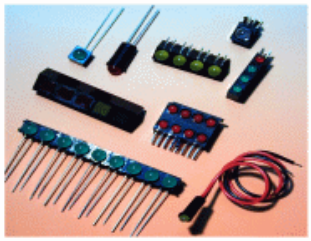
Şekil 9.8 Süper Flux LED Lamp 'lari, Şekil 9.9 ise Axial başlıklı LED Lamp'ları göstermektedir.



Şekil 9.10

Şekil 9.11

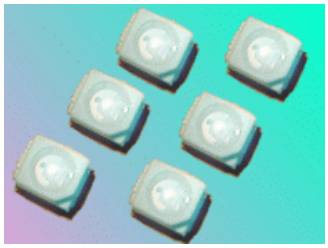
Şekil 9.10 plastik kaplı Axial LED Lamp' ları göstermektedir. Şekil 9.11 IR/PT/PO LED Lamp 'ları göstermektedir.



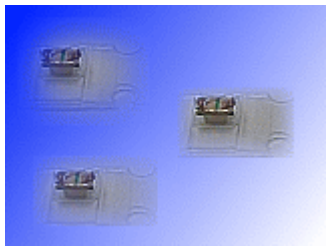
Şekil 9.12

Şekil 9.13

Şekil 9.12 Plastik tutucu içerisinde yer alan LED Lamp 'leri göstermektedir. Şekil 9.13 ise küme halinde LED Lamp 'leri göstermektedir.



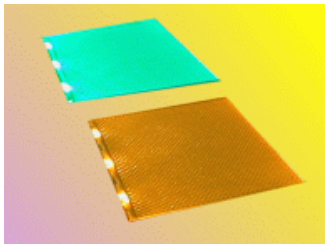
Şekil 9.14



Şekil 9.15

Şekil 9.14 SMD Top LED , Şekil 9.15 ise SMD (PCB) Child LED ' e ait örnekler içermektedir.

Şekil 9.16



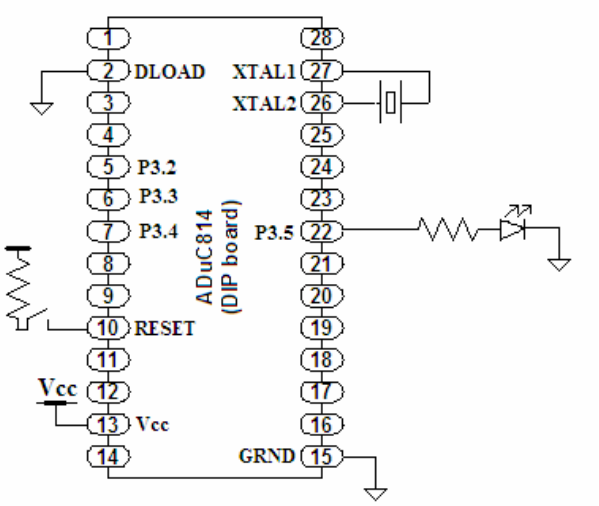
Şekil 9.17

Şekil 9.16 'de BASE LED , Şekil 9.17 ise LED BACK LIGHT örnekleri yer almaktadır.

Şekil 9.18

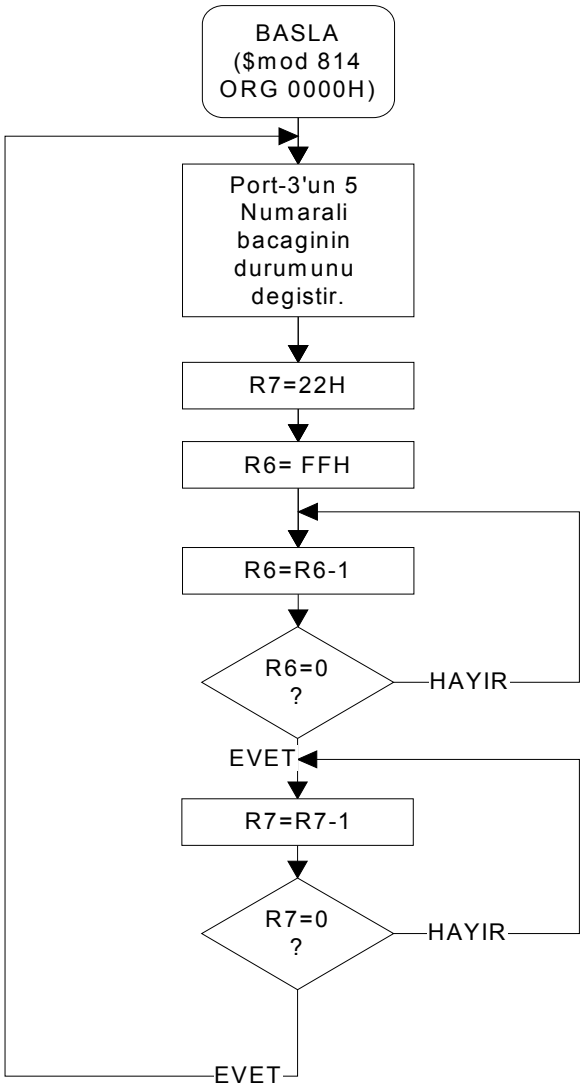
Şekil 9.18 farklı boyutlarda ve digitlere sahip LED DISPLAY örnekleri yer almktadır.

Program Amacı: Bir LED in durumunun 100mS lik periyodlar ile değiştirilmesi.
Devre Bağlantısı:



Akış Diagramı:

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr



Program Kodu:

```
      $MOD814                 ;8052 Tabanlı ADuC ön koşullamalarının mod
                              ;bulunduğu dosyası

      ORG   0000H             ;
                              ;
      JMP   ANA_DONGU         ;
                              ;Kontrol etmek istediğimiz vektör adreslerinin
      ORG   0003H             ;girilmesi
                              ;
      JMP   KESME_INT0        ;
                              ;
      ORG   0060H             ;

;****************************************
ANA_DONGU:
      CPL P3.5                      ;LED in durumunu değiştir
      CALL DELAY                    ;2 MHz clock için 100msn lik
```

```
                                    ;gecikme rutinini çağır

      JMP   ANA_DONGU               ;Döngüyü sonsuz hale getirmek için
                                    ;sürekli başa dön
;*********************************************************************
DELAY:

      MOV   R7,#022H                ;Alt rutinin mantığında 22HxFFH kez işlem
                                    ;yapılarak 100msn lik bir gecikme
                                    ;sağlanması yatar.Birinci çarpan olan R7
                                    ;registerine gereken değer atanır
DLY:

      MOV   R6,#0FFH                ;İkinci çarpan olan R6 değişkenine
      DJNZ  R6,$                    ;gerekli değer atanır ve sıfır olana
                                    ;kadar 1 azaltılıp aynı işlem

                                    ;sırasının aynı satıra atlaması sağlanır


      DJNZ  R7,DLY                  ;R6 nın her sıfır oluşunda FFH değerinin
                                    ; tekrar yüklenip azaltma işleminin R7
                               ; içeriği kadar tekrarlanması bu satırda
                                    ;sağlanır

      RET                           ;Alt rutinden çağrıldığı satıra dönüş.
;***********************************************
      END
```

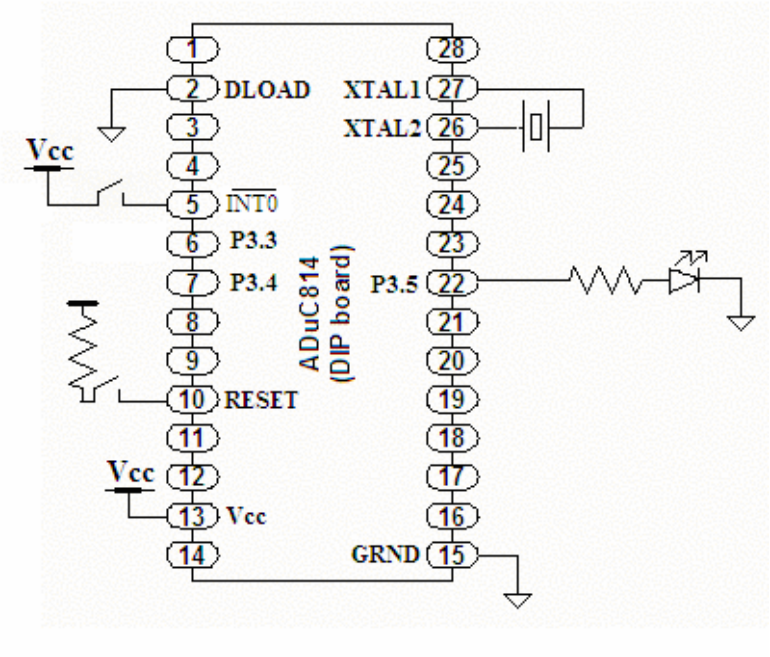
Ek Bilgi: Bu gecikme işleminde 22HxFFH kez beklemenin yanı sıra her komutun icra süreside göz önüne alınarak 2.097152 MHz bir PLL clock yapısıa göre 100msn lik gecikme sağlanmıştır.Diğer clock değerleri için böyle bir rutin hazırlamak veya bu rutini uyarlamak kolay bir işlemdir.Kendi kendinize bu kodu satır satır işlemeniz ve her komutun ayrı ayrı icra süresini toplamanız ve istenen gecikme süresi için R7 ve R6 registerlarına gereken çarpanları eklemeniz yeterli olacaktır.

9.3.2 INT0 dış kesmesi kullanılarak flaşör LED periodunun yükseltilmesi

Ön Bilgi: Delay gecikmesi kullanılarak yaptığımız flaşör LED programının 100msn lik periyodunu arttırmak ve bu arttırmayı donanımsal olarak yapmak için INT0 kesmesini kullanmak yeterli olacaktır.8051 mimari yapısındaki INT0 kesmesi bu programda bir alt rutin olarak karşımıza çıkıyor.Tek yapmamız gereken Delay gecikmesi içerisine üçüncü bir çarpan tanımlamak ve her INT0 kesmesi oluştuğunda bu çarpanı bir arttırmaktır.INT0 kesmesini program içinde aktif ettikten sonra donanımsal olarak bir tuş ile INT0 pinini besleme (5V) gerilimine bağlamak her tuşa basıldığında program akışının kesilip belirtilen kesme alt rutinine gitmesi sağlanır.

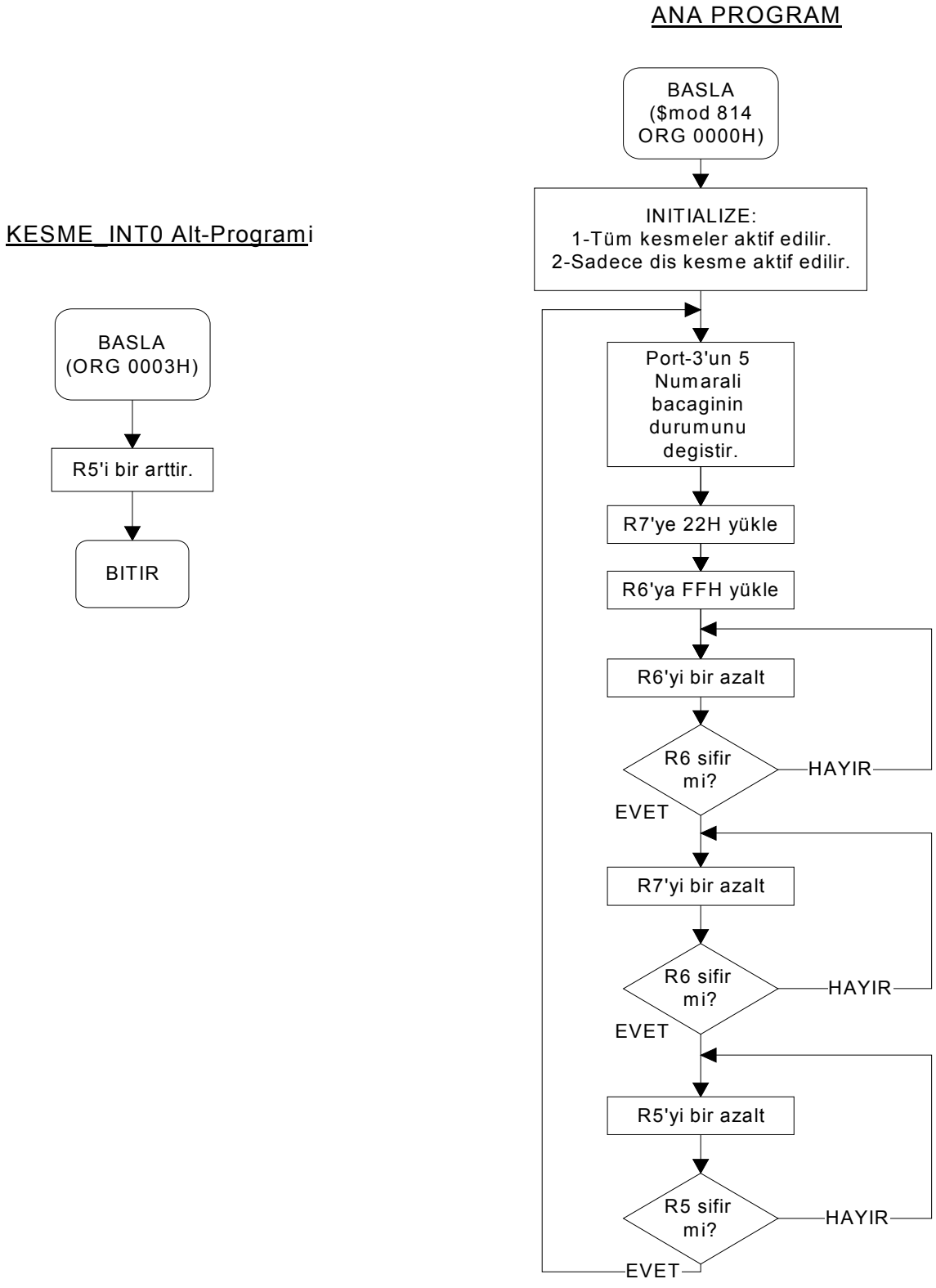
Programın amacı: Her INT0 sinyali alındığında flaşör LED periyodunu 100msn arttırmak.
Devre Bağlantıları:

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr



Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Akış Diyagramı:



Program Kodu:

```
      $MOD814                 ;8052 Tabanlı ADuC ön koşullamalarının
                              ;bulunduğu mod dosyası

      ORG   0000H             ;
                              ;
      JMP   ANA_PROGRAM       ;
                              ;Kontrol etmek istediğimiz vektör adreslerinin
      ORG   0003H             ;girilmesi
                              ;
      JMP   KESME_INT0        ;
                              ;
      ORG   0060H             ;
;*************************************************
ANA_PROGRAM:
;-----------------------------------------------------------------
INITIALIZE:                   ;Ön koşullamaların yapıldığı bölüm

      SETB  EA                ;Kesmeler için gereken izin

      SETB  EX0               ;INT0 için gereken izin

      MOV   R5,#1             ;Periyodun 100msn ile başlaması için gerek ön
                                    ;koşullama
;-----------------------------------------------------------------------
ANA_DONGU:

      CPL P3.5                ;LED in durumunu değiştir
      CALL DELAY               ;2 MHz clock için 100msn lik ;gecikme rutinini
                              ;çağır

      JMP   ANA_DONGU         ;Döngüyü sonsuz hale getirmek için sürekli başa
                              ;dön
;*************************************************************************
DELAY:

      MOV   R7,#022H          ;Alt rutinin mantığında 22HxFFH kez işlem
                              ;yapılarak 100msn lik bir gecikme sağlanması
                              ; yatar.Birinci çarpan olan R7 registerine
                              ; gereken değer atanır
DLY:

      MOV   R6,#0FFH          ;İkinci çarpan olan R6 değişkenine gerekli
      DJNZ  R6,$              ; değer atanır ve sıfır olana kadar 1
                        ;azaltılıp aynıişlem sırasının aynı satıra

      DJNZ  R7,DLY            ; atlaması sağlanır R6 nın her sıfır oluşunda
                              ; FFH değerinin tekrar yüklenip azaltma
                              ; işleminin R7 içeriği kadar tekrarlanması bu
                              ; satırda sağlanır
      DJNZ  R5,DELAY          ;R5 içeriği kadar DELAY rutinin
                              ; tekrarlanmasını sağlayan komut
      RET                     ;Alt rutinden çağrıldığı satıra dönüş.

;*************************************************************
KESME_INT0:
```

```
      INC   R5                      ;R5 Çarpanının bir arttırılması

      RETI                          ;Kesmeden geri dönüş
;*************************************************************
      END
```

Ek Bilgi: Kesme programları 8051 mantığıyla bire bir olduğundan 8051 uygulamaları bölümünden bilgi alabilirsiniz.Bu komut örneğinde kullanılan R5, R6, R7 registerları başka bölümlerde de gerekebileceğinden yerlerine Internal RAM üzerinden tanımlanmış sabit değişkenler kullanmak faydalı olacaktır.

9.3.3 Delay Alt Rutinlerinin İstenilen Uzunluklarda Hazırlanması

9.3.3.1 1 Saniye lik gecikme

Ön Bilgi: Bu uzunluktaki bir gecikmeyi hazırlamak için 100msn lik gecikme programımıza dış kesme örneğinde kullandığımız gibi üçüncü bir çarpanı ekleyip içeriğini 10 olarak sabitlemek yeterlidir. Ancak her rutin çağırıldığında bu içeriği tazelemek unutulmamalıdır.

Program Kodu:

```
;*********************************************************
DELAY:
      MOV   R5,#10

DLY0:
      MOV   R7,#022H          ;Alt rutinin mantığında 22HxFFH kez işlem
                              ;yapılarak 100msn lik bir gecikme sağlanması
                              ;yatar.Birinci çarpan olan R7 registerine
                              ; gereken değer atanır
DLY1:
      MOV   R6,#0FFH          ;İkinci çarpan olan R6 değişkenine gerekli
      DJNZ  R6,$              ; değer atanır ve sıfır olana kadar 1
                        ; azaltılıp aynıişlem sırasının aynı satıra
                              ; atlaması sağlanır

      DJNZ  R7,DLY            ;R6 nın her sıfır oluşunda FFH değerinin
                              ;tekrar yüklenip azaltma işleminin R7 içeriği
                              ;kadar tekrarlanması bu satırda sağlanır

      DJNZ  R5,DLY0           ;R5 içeriği kadar DELAY rutinin
                              ;tekrarlanmasını sağlayan komut
      RET                     ;Alt rutinden çağrıldığı satıra dönüş.
;*********************************************************
```

9.3.3.2 10 MSN lik gecikme

Ön Bilgi:Üçüncü çarpanlarla şimdiye kadar kolayca değiştirdiğimiz 100msn lik rutin bu noktada biraz daha değiştirilmesi zor bir hale gelecektir.Çünkü üçüncü bir çarpan olarak 1/10 gibi bir değer atamak tam sayılarla çalışan bir işlemci mantığına aykırıdır.Bunun yerine ilk iki çarpandan birinin değeri 1/10 una en yakın tam sayı değer ile değiştirilmelidir.Burada 22H ve FFH değerlerinin Decimal karşılıklarına göz atalım.22H=34 ve FFH=255.Bu sayıların 1/10 u ise 3.4 ve 25.5 tir.Bu noktada 3.4 ü 3 olarak kabul etmek ve 22H değerinin yerine 3 yazmak mantıklı olur.

Program Kodu:

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

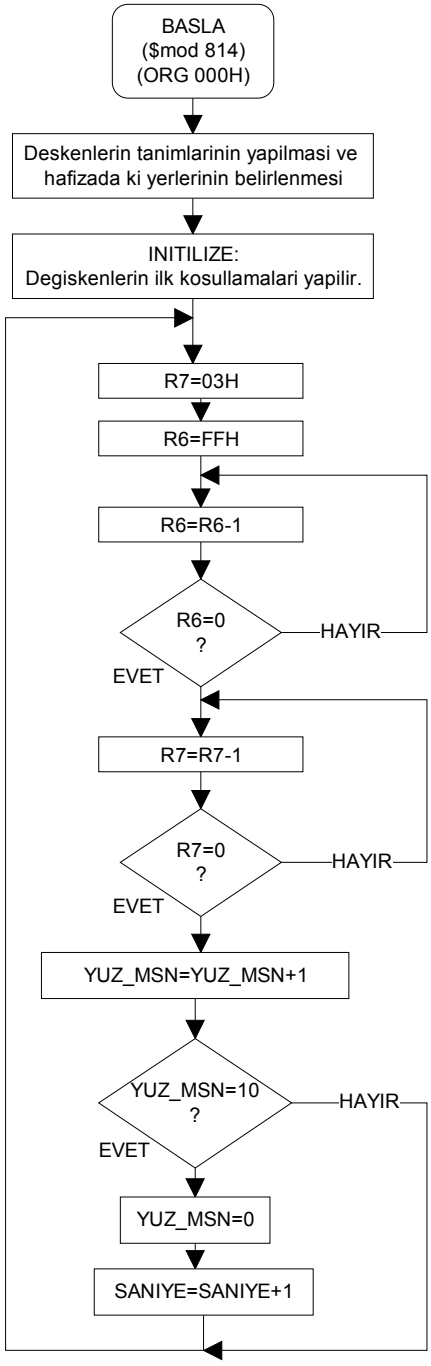
```
;*******************************************
DELAY:

      MOV   R7,#03H           ;Alt rutinin mantığında 03HxFFH kez işlem
                              ;yapılarak 100msn lik bir gecikme sağlanması
                              ; yatar.Birinci çarpan olan R7 registerine
                              ; gereken değer atanır
DLY:

      MOV   R6,#0FFH          ;İkinci çarpan olan R6 değişkenine gerekli
      DJNZ  R6,$              ;değer atanır ve sıfır olana kadar 1 azaltılıp
                              ;aynıişlem sırasının aynı satıra atlaması
                              ;sağlanır


      DJNZ  R7,DLY            ;R6 nın her sıfır oluşunda FFH değerinin
                              ; tekrar yüklenip azaltma işleminin R7 içeriği
                              ;kadar tekrarlanması bu satırda sağlanır


      RET                     ;Alt rutinden çağrıldığı satıra dönüş.
;*************************************************
```

Ek Bilgi: Bu tarz bir yaklaşım kusursuz zamanlama ile sonuçlanmaya bilir ancak gecikme alt rutinleri ile zamanlama yapmak zaten hiçbir gerçek hayat tabanlı uygulamada verimli olmayacaktır. Bunun yerine kusursuza daha yakın olan timer özelliği kullanılabilir.İlerleyen uygulamalarda bu konuya geri döneceğiz.

9.3.4 Delay Alt Rutinini Kullanarak Geçen Süreye Göre "YUZ_MSN" ve "SANIYE" isimli değişken içeriklerinin arttırılması.

Ön Bilgi: Sadece bir gecikme rutini kullanılarak periyodik bir zaman aralığı ölçüldüğünde bir birinin katları olan zaman değerleri hesaplanabilir.Her on milisaniyelik period sonunda 1 artan bir değişken; 10 kez arttığında her yüz milisaniyede 1 artan bir değişkeni etkilerse çok basit bir saat örneği ortaya çıkar.Bu programda 100 msnlik bir delay programının döngü içinde on kere çağırılması sonucu 1 arttırılan SANIYE değişkeninin kullanımını inceleyeceğiz.

Akış Diagramı:



Program Kodu:

```
      $MOD814                       ;8052 Tabanlı ADuC ön koşullamalarının
                                    ;bulunduğu mod dosyası
DEFINITIONS:

      YUZ_MSN     DATA        06FH  ; SANIYE ve YUZ_MSN  değişkenlerini
internal RAM
      SANIYE      DATA        06EH  ; üzerinde  uygun gördüğümüz bölgeye
```

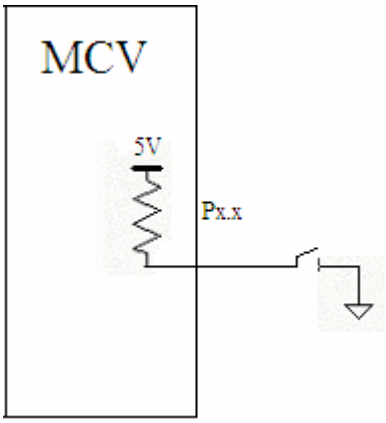
```
                                    ; atıyoruz


;-------------------------------------
      ORG   0000H             ;
                              ;
      JMP   ANA_PROGRAM       ;
                              ;Kontrol etmek istediğimiz
                              ;vektör adreslerinin girilmesi
      ORG   0060H             ;
;*********************************
ANA_PROGRAM:

INITIALIZE:                   ;Ön koşullamaların yapıldığı bölüm

      MOV   YUZ_MSN,#0H       ;Değişkenlerimizin gösterdiği adreslerde önceden
      MOV   SANIYE,#0H        ;bulunan  herhangi bir değeri temizliyoruz

;-----------------------------------------------------
ANA_DONGU:

      CALL  DELAY

      JMP ANA_DONGU
;***************************************
DELAY:

      MOV   R7,#03H           ;Alt rutinin mantığında 03HxFFH kez işlem
                              ;yapılarak 100msn lik bir gecikme sağlanması
                              ;yatar.Birinci çarpan olan R7 registerine
                              ; gereken değer atanır
DLY:

      MOV   R6,#0FFH          ;İkinci çarpan olan R6 değişkenine gerekli
      DJNZ  R6,$              ;değer atanır ve sıfır olana kadar 1 azaltılıp
                              ;aynıişlem sırasının aynı satıra atlaması
                              ;sağlanır

      DJNZ  R7,DLY            ;R6 nın her sıfır oluşunda FFH değerinin
                              ; tekrar yüklenip azaltma işleminin R7 içeriği
                              ;kadar tekrarlanması bu satırda sağlanır

      INC   YUZ_MSN           ;Delay sonucunda 100msn geçmiş olacağı için
                              ; YUZ_MSN değişkenini bir arttır
      MOV   A,YUZ_MSN         ;Taşmayı kontrol etmek için YUZ_MSN içeriğini
      CJNE  A,#10,DON_DLY     ;A'ya ata A yı 10 ile karşılaştır eşit değilse

                              ;DON_DLY etiketinden çık eğer eşitse altsatıra
      MOV   YUZ_MSN,#0        ;geç Taşan YUZ_MSN değişkenini sıfırla

      INC   SANIYE            ;10 Kez yuz msn artması sonucu saniyeyi 1
                              ; arttır.
      RET                     ;Alt rutinden çağrıldığı satıra dönüş.
;*********************************************
      END
```

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Ek Bilgi: Bu örnek daha önceden de belirtildiği gibi gerçek zamanlı uygulamalarda kullanılmak için yetersiz kalabilir.Ancak zamanın akış işleminin kavranmasında çok önem kazanmaktadır. Saniye değişkeni her 60 olduğunda dakikanın 1 artması ve her 60 dakikada saat içeriğinin değişmesi zamanlı işlemlerde ve zamanın görüntülenmesini gerektiren uygulamalarda kolaylık sağlayacaktır.

9.3.5 Bir I/O Pinine Bağlı Bir Tuşa Basıldığı Sürece Işık Veren LED sürülmesi

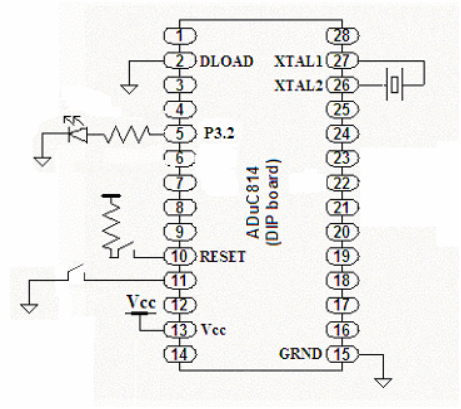
Ön Bilgi:8051 tabanlı ADuC 814 Mikrokonverterımızın port pinlerini dışarıdan kontrol edebiliriz.Bu port pinleri standart olarak bir Pull-Up direnç kullanılarak aktif-high(5V) konumuna getirilmiştir.Bu pinler programla veya donanımsal olarak sıfırlanmadığı sürece logic 1 kalır ve üzerlerinde 5V gerilim gösterir.Bu anlatım aşağıdaki şemada daha iyi görüntülenebilir.



Yandaki şekilde basit olarak bir port pinin içeriden Pull-Up dirençle yükseltilmiş halini görüyoruz.Tuşa basıldığında eğer içeriden önceden "0" olarak koşullandırılmamışsa bu port pininden "0" değeri okunur.

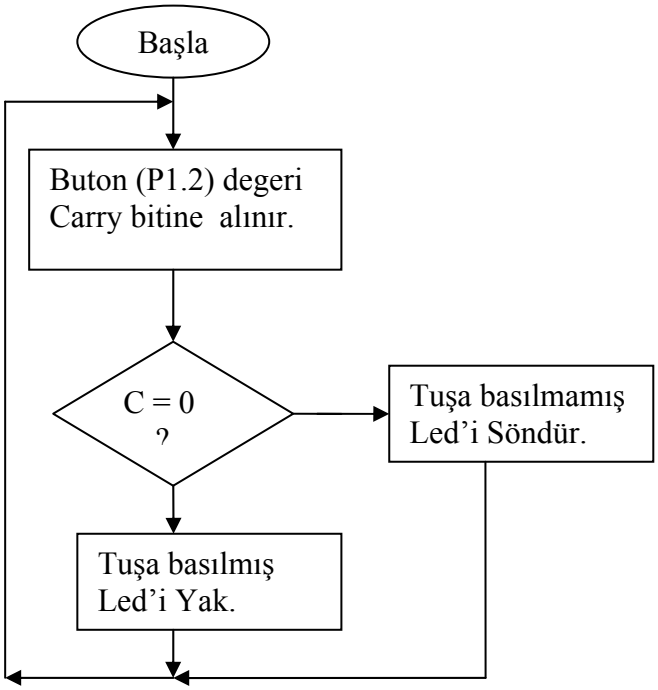
Programın Amacı: Donanımda bir port pinine bağlı bulunan bir tuş ile diğer bir port pinine bağlı bir LEDin kontrolünü sağlamak.

Devre Bağlantıları:



Yukarıda verilen devre şemasında da görüldüğü gibi; Led 'in yakıp söndürme kontrolünü sağlayan buton Port 1 'in 2. bitine bağlanmıştır. Tuş basılı değilken P1.2 'da 5V gerilimden ötürü logic 1 değeri görülmektedir. Tuşa basıldığı takdirde; toprağa bağlanılacağı için logic 0 değeri görülecektir.Led 'in yanması için ise Port 3'ün 2. biti logic 1 değerini almalıdır.

Akış Diyagramı :



Program Kodu:

```
.......
BUTON_KONTROLLU_LED_YAKMA:
BUTON_KONTROL :

      MOV C , P1.2                ; Tuşun bağlı olduğu port pininin
                                  ; değerini C içine taşınır.

      JC   TUS_BASILI_DEGIL       ; C nin değeri kontrol edilir.
                                  ; 1 ' se basılı degildir
```

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

```
                                    ; TUSA_BASILI_DEGIL  satırına gidilir.
                                    ; 0'a eşitse tusa basılmıştır.
      SETB  P3.2                    ; Led'i yakmak için P3.2 logic 1 yapılır.

      JMP  BUTON_KONTROL            ; Butonu tekrar kontrol için başadönülür.

TUS_BASILI_DEGIL:

      CLR   P3.2                    ; tuş basılı değil ise Led `, söndür.
                                    ; P3.2  logic 0 yapılır.
      JMP  BUTON_KONTROL            ; Döngüyü sonsuz hale getirilir.
```

Buton kontrolu ile Led yakıp söndürme programı bu şekild kontrollerle gerçekleştirilebilir. Ayrıca ayı programı aşağıdaki şekilde de gerçekleştrimek de mümkün.

```
...
BUTON_KNTRLLU_LED_YAKMA:

      MOV         C,P1.2            ;Tuşun bağlı olduğu port pininin değerini
                                    ;C içine taşıyoruz
      CPL         C                 ;Tuşa basılıyken LED in ışık vermesini
                                    ;istediğimizden C yikomplimentini
                                    ;almalıyız çünkü tuşa bastığımızda P1.0
                                    ;den 0 okunacaktır bu pin "0" olduğunda
                                    ;P3.2 "1" olmalıdır

      MOV         P3.2,C            ;Son olarak C içeriğini P3.2 içine atıyor
                              ;ve LED ışık verdiğini görüyoruz.
      JMP   BUTON_KNTRLLU_LED_YAKMA: ;Döngüyü sonsuz hale getiriyoruz.
```

İkinci blokda verilen kod örneği ilkine göre daha verimlidir. İkinci kısımda mikrodenetleyicilerin mikroişlemcilere göre avantajı olabilecek bir özelliği efektif bir biçimde kullanılmıştır. Mkrodenetleyicilerde portlara bit mertebesinde erişmek mümkündür. Bu da prgramımızın kodunu azaltmakla birlikte akışını da hızlandırmaktadır.

Ek Bilgi: Kitabın 8051 ile ilgili bölümünde de anlatıldığı gibi portların içindeki LATCH yapısı nedeniyle bir Port pininden okuma ancak o port pinin yazılımsal olarak lojik "1" değerinde koşullanmasıyla yapılabilir.İşin gerisi donanıma kalır.Dikkat edilmesi gereken bir diğer nokta ise;

```
      MOV   Px.x,Py.y
```

benzeri bir komutun assembler tarafından hatalı olarak kabul edilir.Bu sebeple port pinleri arasında aktarma C üzerinden yapılmalıdır.

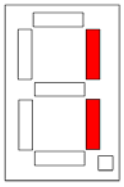
9.3.6 Tek haneli 7-Segment LED Display Sürülmesi

Ortak Anot Bacaklı: LEDlerin anot bacakları tek bir bacak olarak dışarıya verilmiştir. Bu bacak gerilim kaynağına bağlanarak işleme sokulabilir.Bu durumda mikrodenetleyiciden çıkışları Aktif-Low olarak alıp,segmentleri tek tek yakabiliriz.

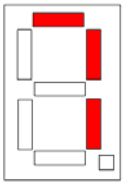
Ortak Katot Bacaklı: LEDlerin katot bacakları tek bir bacak olarak dışarıya verilmiştir. Bu bacak toprağa bağlanarak işleme sokulabilir.Bu durumda mikrodenetleyiciden çıkışları Aktif-High olarak alıp,segmentleri tek tek yakabiliriz.

7-Segment displayın her segmenti bir led olarak düşünürseniz, bir LEDi 5V gerilim ile süremiyorduk, bunun için LED den önce bir 330 Ω'luk direnç yardımıyla LED üzerinden 10mA geçmesini sağlıyorduk. Aynı şekilde 7-Segment displayi sürerken ortak olmayan her bacağa bir direnç(330 Ω) bağlamamız gerekmektedir.

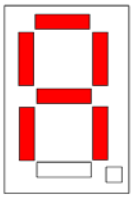
Her bacağa bir direnç yerine ortak anot/katod bacağına bir direnç koymamamızın nedeni ise, böyle bir bağlantıda her segment için 10mA akım gerekeceğinden yanan segment sayısı arttıkça gereken akım değeri artacaktır. Bu akım değerinin artması bağladığımız direnç üzerindeki gerilim düşümü artacağından, yanan segment sayısı arttıkça direnç üzerindeki gerilim düşümü artacaktır. Bu nedenle yanan segmentlerin parlaklığı azalacaktır.Bu her sayıda farklı parlaklık gözleneceğinden başarısız bir durum oluşacaktır.



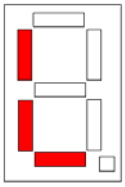
```
MOV  P1,# 11111001B        ;  "1" görüntüsü 7-segment displayde görünür.
```



```
MOV  P1, #11111000B        ; "7" görüntüsü 7-segment displayde görünür.
```



```
MOV  P1, #10001000B        ;"A" görüntüsü 7-segment displayde görünür.
```

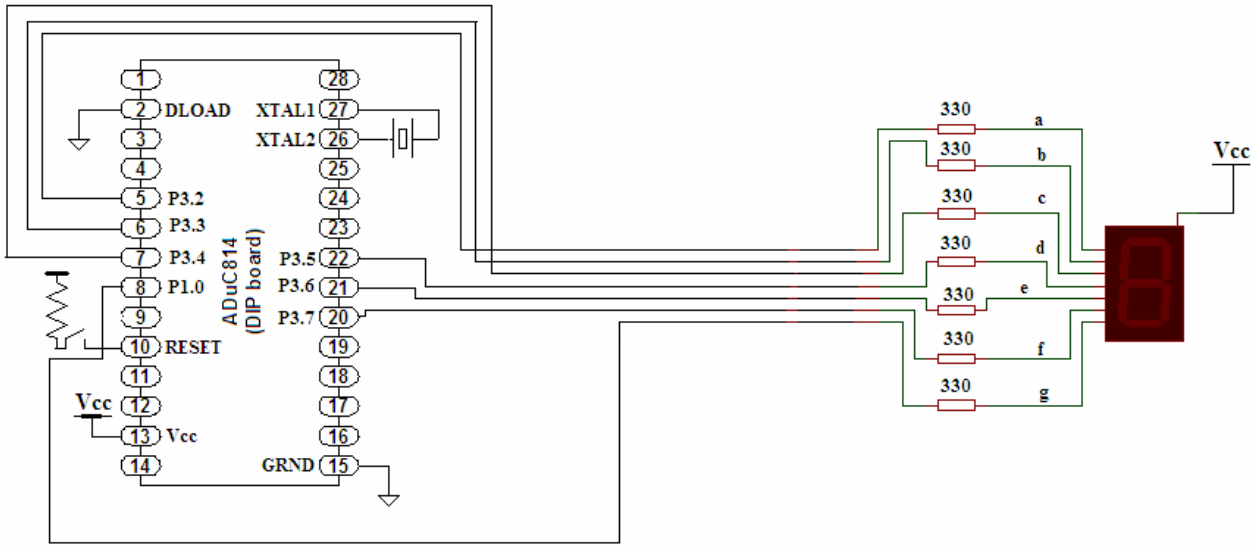


```
MOV  P1, #11000111B        ; "L" görüntüsü 7-segment displayde görünür.
```

Bu çeşit bir 7-Segment sürme yöntemi ADuC 814 için oldukça zordur.Sonuç olarak tüm pinlerinden LED sürebildiğimiz bir portumuz bulunmamaktadır.Bu sorun üç ana yöntemle çözülebilir bu bölümde üç yöntemi de inceleyelim.

Yöntem-1-**Portların pinlerini tek tek ayarlamak:**

Devre Bağlantıları:



Not:Bu bağlantı şekilleri kullanım kolaylığı açısından değiştirilebilir.Ancak programında değiştirilmesi unutulmamalıdır.

a=P3.2 b=P3.3 c=P3.4 d=p3.5 e=p3.6 f=p3.7 g=p1.1

Not:Burada kullanılan 7-Segment ortak anotlu olduğundan ışık vermesi istenen segmente lojik "0" uygulanmalıdır.

Program Kodu:

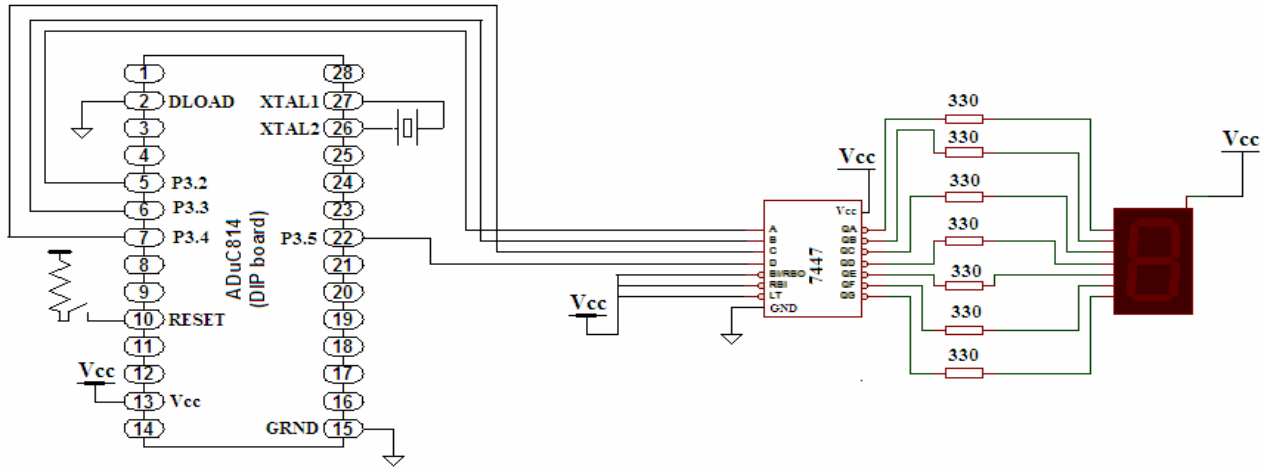
```
;Ekrana "3" Yazdırmak için ;(a,b,c,d,e,f,g)=(0,0,0,0,1,1,0) olarak
koşullanmalı.
      ....
      CLR   P3.2
      CLR   P3.3
      CLR   P3.4
      CLR   P3.5
      SETB  P3.6
      SETB  P3.7
      CLR   P1.0
      ...
;"A" harfi içinse (a,b,c,d,e,f,g)=(0,0,0,1,0,0,0)
      ......
      CLR   P3.2
      CLR   P3.3
      CLR   P3.4
      SETB  P3.5
      CLR   P3.6
      CLR   P3.7
      CLR   P1.0
      …….
```

Bu şekilde ekranda istenen sabit şekil gösterilebilir..

Yöntem-2-**7447 BCD to 7-Segment Dönüştürücüsüyle**

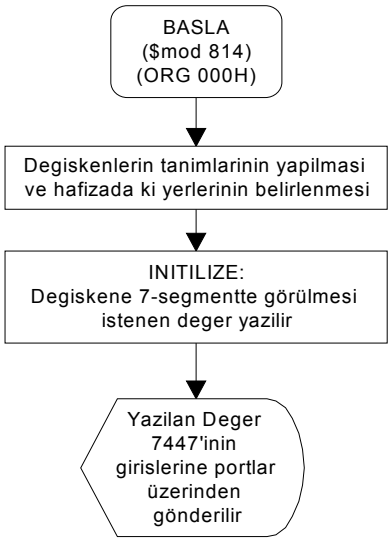
Önemli:*Bu entegrenin özelliği 3 pinine atadığınız "1" ve "0" değerini okuma pinlerinden BCD olarak algılayıp sürme port pinlerine uygun sayı değerini atamasıdır.Örneğin ekranda "8" değerini görüntülemek için A B C D (bazı modellerde A0 A1 A2 A3 olarak adlandırılabilir) okuma pinlerine "(1,0,0,0)=(D,C,B,A)" değerlerini atamak yeterlidir. Bu şekilde ekranda 0-15 arası sayılar görüntülenebilir.*

Devre Bağlantıları:



P3.2=A P3.3=B P3.4=C P3.5=D

Akış Diagramı:



Program Kodu:

```
      $MOD814                    ;8052 Tabanlı ADuC ön koşullamalarının
                                 ;bulunduğu mod dosyası
DEFINITIONS:
```

```
      HANE_BIR_BUF      DATA  06FH
                                  ;HANE_BIR_BUF  değişkenini internal
                                  ; RAM üzerinde  uygun gördüğümüz bölgeye
                                  ; atıyoruz
;--------------------------------------
      ORG   0000H                 ;
                                  ;
      JMP   ANA_PROGRAM           ;
                                  ;Kontrol etmek istediğimiz vektör
                                  ;adreslerinin girilmesi
      ORG   0060H                 ;
;*********************************
ANA_PROGRAM:

INITIALIZE:                       ;Ön koşullamaların yapıldığı bölüm

      MOV   HANE_BIR_BUF,#00001000B ;Hanede görünmesini istediğimiz sayının
BCD
                                  ;değerini bu değişkenin alt 4-bitine
                                  ;yazıyoruz
;-------------------------------------------------
ANA_DONGU:

      CALL  HANEYE_YAZ

      JMP ANA_DONGU
;***************************************
HANEYE_YAZ:

      MOV   A,HANE_BIR_BUF          ;Her biti gereken porta yazabilmek üzere
                            ;yapacağımız RRC işlemini gerçekleştirmek
                        ;için değişkeni A içine atıyoruz
                                  ;ilk bit
      RRC   A
      MOV   P3.2,C
                                  ;ikinci bit
      RRC   A
      MOV   P3.3,C
                                  ;üçüncü bit
      RRC   A
      MOV   P3.4,C
                                  ;dördüncü bit
      RRC   A
      MOV   P3.4,C

      RET
;*****************************************
      END
```

Önemli-Ek Bilgi:*Bu uygulamada en önemli programlama prensibi yani bir programın ondan istenen dışında hiçbir işlem yapmaması göz önünde bulundurulmuştur.Bu nedenle diğer port pinleri etkilenmeden istenen port pinlerinin ayarlanması sağlanmaya çalışılmıştır.*

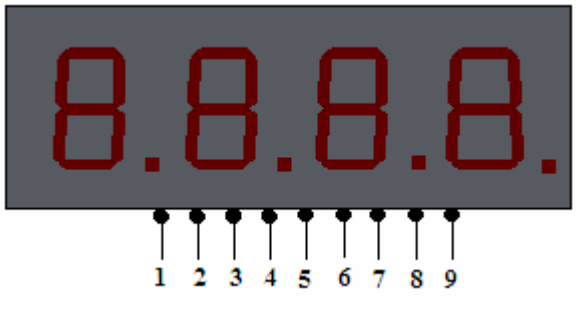
*Yöntem-3-***74595 Port Entegresi**

Bu entegrenin kullanım özellikleri bu konudan çok daha geniş olduğundan bu uygulama ileriki bir bölümde anlatılmıştır.

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

9.3.7 Intelligent 7-Segment Display Sürülmesi

Ön Bilgi: Intelligent Display akım sürebilen port pini sıkıntısı çektiğimiz uygulamalarda kullanışlı özellikleri olan bir üründür.Bu display sadece 3 pin kullanarak 4x7-Segment hanesini istediğimiz görüntüyle doldurabilmemizi sağlar.Bu bölümde bir Intelligent Display' in sürülürek ekranda "1234" gibi sabit bir sayının yazdırılması anlatılmaktadır.



Bacakların Özellikleri:
1 ve 2 nolu bacaklar intelligent display ile LED sürmek için kullanılır.
3 nolu bacak Clock giriş bacağıdır.
4 nolu bacak Data Enable bacağıdır.
5 nolu bacak Data ınput bacağıdır.
6 nolu bacak V_{CC} bacağıdır,ıntelligent displayın beslemesidir.
7 nolu bacak Brightness Control bacağıdır.
8 nolu bacak V_{DD} (toprak) bacağıdır.
9 nolu bacak V_{LED} bacağıdır. Displaydeki ledlerin beslemesidir.

Intelligent Display Çalışma mantığı:

İntelligent display her hangi bir kaynak ile üretilen "clock pulse" mantığına çok benzeyen data sinyalleri sayesinde mikro denetleyici ile haberleşir.Normal clock mantığından farkı ise gönderilecek olan dalga sinyallerindeki lojik "1" ve "0" lar her hangi seçilen bir port pininden kullanıcı tarafından tek tek üretilir.İntelligent led display her bir mesajda hangi hanelerde hangi ledlerin yakılacağını okur.

Her bir mesaj 36 darbeden oluşur "Clock" pini olarak atadığımız portun her iki darbe üretişi arasında "DataIn" olarak kullandığımız port taki bilgi Display modülüne gönderilir.Bu 36 darbelik mesajın ilk darbesi START bit olarak kabul edilir.Bu bit gönderildiği anda Display mesaj okumaya hazır hale gelmiş olur.Diğer 35 bit içeriğinde ise her 4 haneye gönderilen 8 bit için (7 tane yazılan ve 1 tane de nokta ayracı için) 4x8=32 darbe gönderilir.36 lık periyodu tamamlamak amacıyla gönderilen 2 darbe ve mesajı sonlandıran STOP bit de gönderildiğinde ekrana istenilen görüntü gönderilmiş olur.

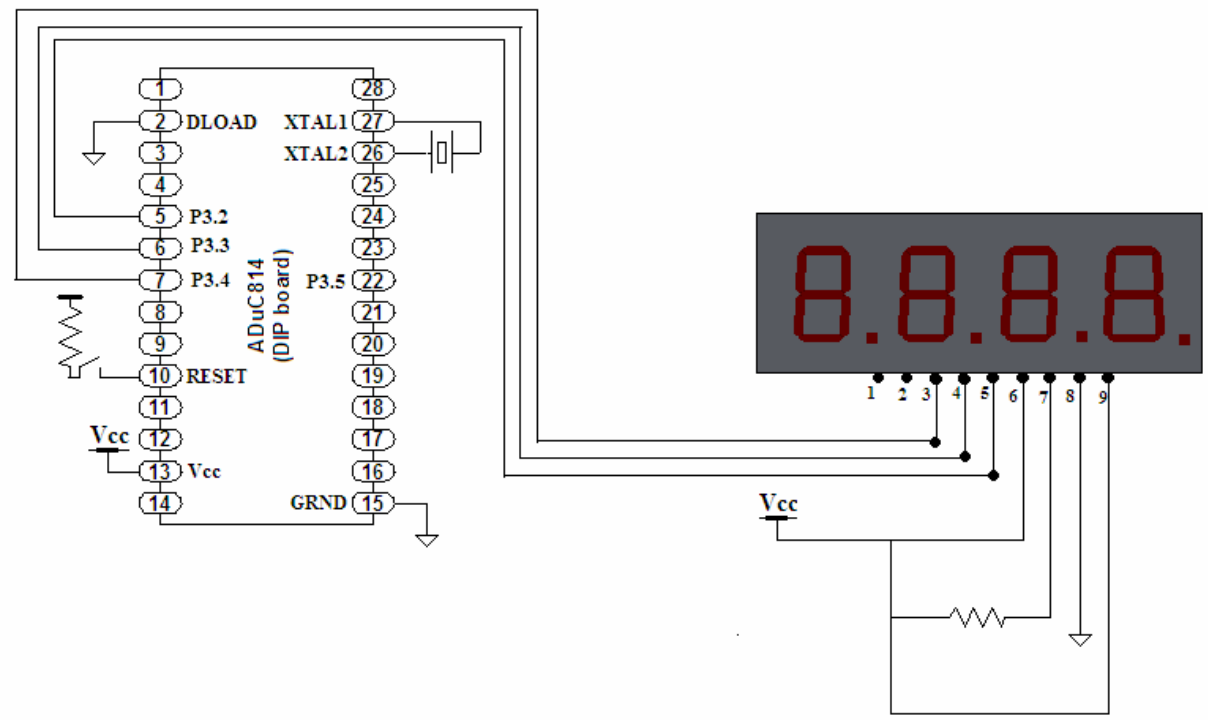
Bu nokta dışında normal bir SEVEN SEGMENT LED DISPLAY'den hiçbir farkı yoktur.Önceki deney örneğinde gösterdiğimiz tek haneli SEV_SEG yakma programında a-g arası isimlendirilmiş olan LED leri gerekli sayıları uygun port bitinde lojik "0" ve "1" yaparak çalıştırıyorduk burada ise yazdığımız sayıları BIT-BIT gönderme işlemi sayesinde 36 darbelik mesajın içine yazacağız.Bu işlem kod örneğinde daha detaylı olarak açıklanmıştır.

Belirtilmesi gereken önemli bir nokta yazılacak olan haneleri belirten lojik değerlerin hazırlanmasıdır. Bu konu SEV-SEG 'lerin ortak anot veya katot özelliklerine göre sayıların hazırlanmasıdır.Int-Led Display'de sayılar ortak katotlu bir SEV-SEG'e göre hazırlanır.Bu duruma örnek olarak ekrana "0" yazdırmak istiyorsak gerekli hanenin bufferına;

```
MOV   BIRLER_BUF,#00111111B
```

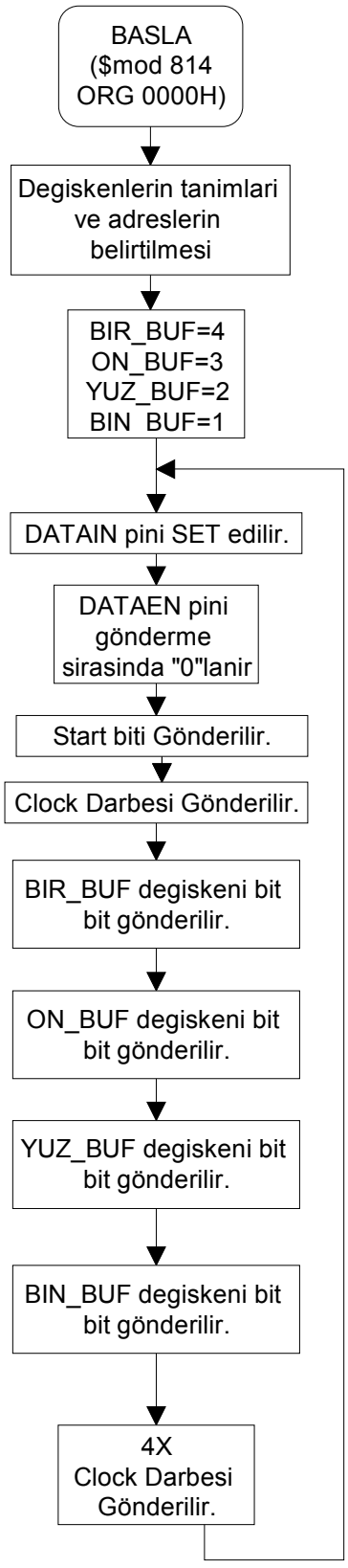
Şeklinde gerekli sayı işlenmelidir.

Bu tür bir DISPLAY modülünü sürecek olan ADuC814 mikrodenetleyici bağlantı şekli aşağıdaki gibidir.



Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Akış Diagramı:



Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Program Kodu:

```
      $MOD814

      BIR_BUF     DATA        07CH
      ON_BUF      DATA        07BH
      YUZ_BUF     DATA        07AH
      BIN_BUF     DATA        079H
      SAYAC       DATA        078H

      CLOCK       EQU         P3.2
      DATAIN      EQU         P3.3
      DATAEN      EQU         P3.4

      CSEG

      ORG   0000H

      JMP   MAIN

      ORG   0060H

MAIN:

      MOV   BIR_BUF,#4              ;Ekrana yazdırılacak "1234" sayısı
      MOV   ON_BUF,#3               ;bufferlara atanıyor
      MOV   YUZ_BUF,#2
      MOV   BIN_BUF,#1

      MOV   SAYAC,#0                ;Gerekli sayac sıfırlanıyor

                                    ;Int display sıfırla
      MOV   SAYAC,#36

RST_INT_DSP:

      CALL  CLOCK_GETIR

      DJNZ  SAYAC,RST_INT_DSP


;*********************************************
ANA_DONGU:

      CALL  DISPLAY_SUR

      JMP   ANA_DONGU

;*********************************************
CLOCK_GETIR:
      SETB  CLOCK
      CLR   CLOCK
      RET

;*********************************************
DISPLAY_SUR:
      CLR   DATAEN
```

```
      SETB  DATAIN
      CALL  CLOCK_GETIR

                                    ;Binler basamağı
      MOV   A,BIN_BUF
      CALL  BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

                                    ;Yüzler basamağı
      MOV   A,YUZ_BUF
      CALL  BIT_BIT_GONDER
      SETB  DATAIN
      CALL  CLOCK_GETIR

                                    ;Onlar basamağı
      MOV   A,ON_BUF
      CALL  BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

                                    ;Birler basamağı
      MOV   A,BIR_BUF
      CALL  BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

      CLR   DATAIN
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      SETB  DATAEN

      RET
;*********************************************
BIT_BIT_GONDER:
      MOV   SAYAC,#07
BBG_TUR:
      RRC   A
      MOV   DATAIN,C
      CALL  CLOCK_GETIR

      DJNZ  SAYAC,BBG_TUR
      RET
;*********************************************
END
```

9.3.8 Timer Uygulaması (Periodik Flaşör LED)

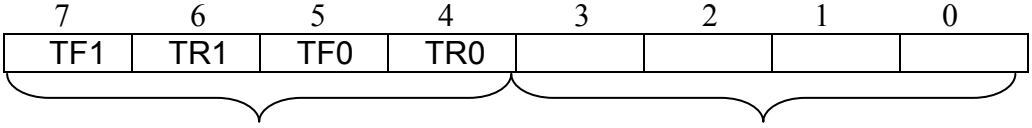
Ön Bilgi: Nano saniyelerle işlemlerini gerçekleştiren bir mikrodenetleyici kullandığımızda zamanlamanın hassaslığı çok önemlidir; bu nedenle mikro denetleyiciler kendi clock frekanslarına göre zamanı tutabilen Timer donanımlarına sahiplerdir.Bu konu 8051 Donanım Özellikleri bölümünde daha detaylı anlatılmıştır.Bizim için gerekli olan kısım ise ADuC 814 kullanılan bir uygulamada gerekli registerların nasıl ayarlandığı ve sayıcı SFR' lerin ilk değerlerinin nasıl doldurulduğu konusudur.

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Timer/Counter çeşitli olaylar sonucu artan bir sayac değişkenidir.Her zaman pozitif yönde değişir.Counterların çalışma mantığından çok farklı olarak her bir makine döngüsünden sonra 1 artar.Her bir makine çevirimi belirli bir kristal darbesine eşit olduğundan diyebilirizki mikrodenetleyicimizde kullandığımız kristalin frekansı hesaplamalarımızla doğrudan ilgilidir.

Timerları kontrol eden iki SFR ,TCON ve TMOD'dur.Bu registerlar programın initialize kısmında bir kez ayarlanır.

TCON:

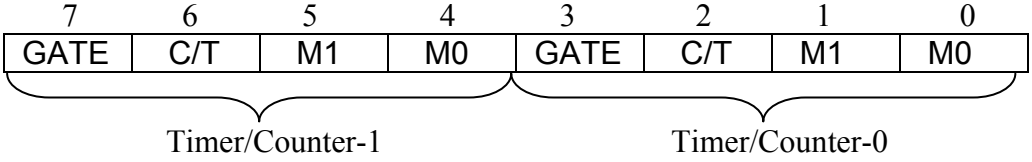


TF1: Timer-1 taşma bayrağı. Sayıcı FFFFH değerinden 0000H değerine atladığında bir taşama bayrağı oluşturur.
TR1: Timer-1'i çalıştırma bitidir. Bu bit "1" yapıldığında timer-1 saymaya başlar.

TF0: Timer-0 taşma bayrağı. Sayıcı FFFFH değerinden 0000H değerine atladığında bir taşama bayrağı oluşturur.
TR0: Timer-0'i çalıştırma bitidir. Bu bit "1" yapıldığında timer-0 saymaya başlar.

Örnek uygulamamızda kullanıcı TCON SFR sine dışardan bir etki edemeyecektir.Sadece istenilen timerin TRx bitini set ederek timeri saydırmaya başlatabilir.Ayrıca TFx bitleri bir kesme alt-rutini için gerekli olacaktır.

TMOD :



Şekilde görüldüğü gibi TMOD'un alt dört biti Timer/Counter-0 'ın kontrol bitlerini içermektedir.Aynı register'ın üst dört biti ise Timer/Counter-1'in kontrol bitlerini içermektedir.

GATE: Gate biti kontrol biti "1" olduğunda Timer/Counter-0 veya 1 in çalışabilmesi için INT0 dış kesme girişinin 1 olması yeterli olacaktır. Sonuç olarak INT0 biti ile GATE'ın tersi AND kapısı ile bağlıdır. Bu şekilde INT0 ile Timer/Counter dışarında kontrol edilir.

C/T: Timer veya counter seçme bitidir. Eğer bu bit "1" seçilirse counter modunda T_n v girişiyle chip dışında oluşan bir olayı sayabilir, "0" seçilirse timer modunda makine periyoduna göre zamanlayıcı olarak çalışır.

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Çalışma modları:

	M1	M0	
MOD0	0	0	TL_x 5 bit ön bölücü, THx 8-bit zamanlayıcı
MOD1	0	1	TL_x ve THx ardışık olarak 16-bit zamanlayıcı
MOD2	1	0	8-bit otomatik yüklemeli çalışma.
MOD3	1	1	Timer-0, TL0 ve TH0 registerları ayrı birer 8-bitlik zamanlayıcı olarak kullanılır.TL0 kontrolü,Timer-0 kontrol kısmından ;TH0 kontrolü Timer-1 kontrol kısmından gerçekleştirilir.

Bu bölümde de gereken değerler TMOD un içine atanmalıdır.Ancak hangi timeri kullanacağımız burada belirtildiğinden dikkatli olunmalıdır.

Timer tamamen bir kesme alt-rutini halinde çalıştırılmalıdır.Programların çalışma mantığına ve sizin projenizin gerektirdiği zaman periyodunda bir kesme yaratılmalı Timer ISR(interrup service routine) içerisinde ele alınmalı gerekli değişiklikler yapıldıktan sonra da geri dönülmelidir. Seçilen moda göre timerin interrupt oluşma şekilleri de değişir.Örneğin Mod1 için 16 bitlik bir sayac gibi düşünülmelidir.Bu sayac bizim içine atadığımız ilk değerden saymaya başlayarak,65536 sayısına(FFFFH) geldikten sonra bir kez daha artarsa sıfırlanır ve kesme meydana gelir.Sonuç olarak istediğimiz süreyi ayarlamak tamamen THx ve TLx içine atayacağımız sayılarla ilgilidir. Peki bu sayılar nasıl hesaplanır?

Timerın her makine çeviriminde 1 artan bir sayac olduğunu söylemiştik.Elimizdede f frekansında clock kullanan 8051 tabanlı bir denetleyici olduğunu var sayalım.8051 için her 12 clock darbesi bir makine çevirimine eşittir.f sayısı bir saniyedeki clock darbesi sayısı olduğundan f/12 bir saniyedeki makine çevirimi sayısıdır. Kaç saniyede bir kesme oluşmasını istiyorsak sonucu bir katsayıyla çarparız.Örneğin saniyenin 100 de birinde kesmeye ihtiyacımız varsa 100 e bölmemiz yeterlidir. Çıkan bu sonuç 10 msn ‘de kaç makine çeviriminin bittiğini gösterir.Bu sayı kadar sayan timer değerlerimizin overflow kesmesi üretmesi için bu sayıyı FFFFh sayısından çıkarmamız gerekir.Sonuc olan sayı THx ve TLx içerisine gereken şekilde atanıp timer

```
SETB  TRx
```

gibi bir komutla saymaya başladıktan FFFFh sayısına gelene kadar istediğimiz süre geçer ve kesme oluşur.

İlk kesme için bu uygun bir durumdur ancak bir dahaki makine çevirimde timer registerları 00h değerinden başlayacaklarından süreyi ayarlayan sayılarımız kaybolmuş olur.Bunu engellemek için her kesme oluştuğunda başta da atadığımız değerleri tekrardan atamalıyız.Sonuç olarak kesme servis rutinimizin ilk iki satırı;

```
MOV   TLx,<istenen değer>
MOV   THx,<istenen değer>
```

şeklinde olmalıdır.Önce TLx in doldurulması ayrıca önemlidir çünkü THx FFh makine çeviriminde 1 artar bu yüzden birkaç satır geç doldurulması bir gecikmeye neden olmayacaktır.TLx ise her makine çeviriminde arttığından acilen doldurulması gerekmektedir. Sonuç olarak kesme servis rutininin başında ve initialize kısmında

```
SETB  TRx
```

komutundan önce TLx ve THx içine yazılacak olan sayı şu şekilde hesaplanır:

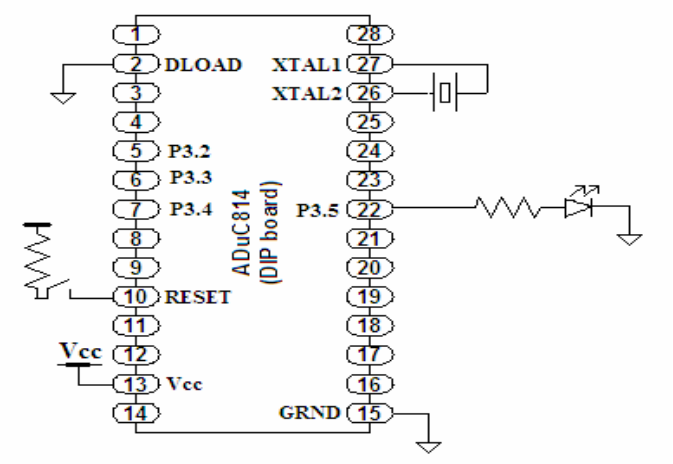
$$FFFFh - \frac{< clockfrekansı >}{12\times < san.katsayı >} = \text{ Timer registerlarına yazılacak olan sayı.}$$

Sayac ve kontrol registerlarımızı nasıl dolduracağımızı öğrendiğimize göre artık uygulamaya geçebiliriz.

Bu uygulama için 2.097152 MHz bir PLL kullandığımızı varsayalım ve her 10msn dolduğunda bir kesme oluşturacak ve her 100msn dolduğunda bir ledin durumunu değiştirecek olan donanım ve yazılımı hazırlayalım.THx ve TLx içerisine doldurulacak sayı

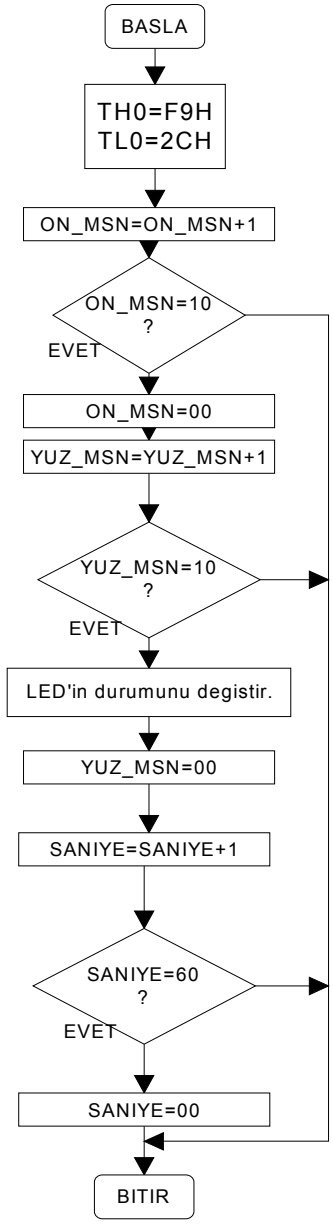
$$FFFFh - \frac{< 2097152 >}{12\times < 0.01 >} = F92C \text{ olmalıdır.}$$

Devre Bağlantıları:

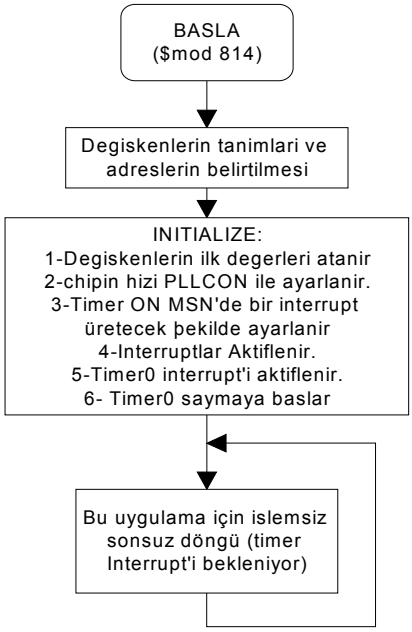


Akış Diagramı:

Timer0 Kesme ServisProgrami



Ana Program



Program Kodu:

```
$MOD814

ON_MSN          DATA  07FH  ;on milisaniyeyi saymak için
                            ;gerekli sayaç tanımlandı.
YUZ_MSN         DATA  07EH  ;yüz milisaniyeyi saymak için
                            ;gerekli sayaç tanımlandı.

SANIYE          DATA  07DH  ;saniyeyi saymak için gerekli sayaç
                            ;tanımlandı.
```

```
      ORG   0000H                  ;
                                   ;
      JMP   ANA_PROGRAM            ;
                                   ;Kontrol etmek istediğimiz vektör
      ORG   000BH                  ; adreslerinin girilmesi
                                   ;
      JMP   TMR0_ISR               ;
                                   ;
      ORG   0060H                  ;

;**************************************
ANA_PROGRAM:

      CALL  INITIALIZE             ;Ön koşulama ve ayarların yapıldığı
                                   ;bölümü çağır.

;---------------------------
ANA_DONGU:                         ;ana programın sonsuz döngüsü

      JMP   $                      ;Bu uygulama için sonsuz işlemsiz döngü
;---------------------------
;**************************************
TMR0_ISR:


     MOV    TL0,#02CH              ; Döngü içerisinde timer-0'a
      MOV   TH0,#0F9H              ; gerekli değerler yüklenir.


      PUSH  ACC                    ;
      PUSH  B                      ;
      PUSH  PSW                    ;Program akışında gerekli olan
                                   ;registerlar stack'e atılarak
      PUSH  DPH                    ;Saklanır.
      PUSH  DPL                    ;

;-------- ON_MSN İŞLEMLERİ

      INC   ON_MSN                 ;ON_MSN değişkeni bir artırılır.
      MOV   A,ON_MSN               ;Karşılaştırma yapmak için ON_MSN'e
                                   ;A'ya aktarılır.

      CJNE  A,#0AH,GERIDON         ;Eğer A (ON_MSN) 10'a ulaştıysa
                                   ;devam et,yoksa GERIDON'e atla.

      MOV   ON_MSN,#00H            ;Bir sonraki döngü için ON_MSN'yi
                                   ;sıfırlayarak  hazır tut.
;----
;--------- YUZ_MSN İŞLEMLERİ

      INC   YUZ_MSN                ;YUZ_MSN değişkenini bir arttır.
      MOV   A,YUZ_MSN              ;Karşılaştırma yapmak için
                                   ;YUZ_MSN'e A'ya aktarılır.
      CJNE  A,#0AH,GERIDON         ;Eğer A (YUZ_MSN) 10'a ulaştıysa
                                   ;devam et,yoksa GERIDON'e atla.
      MOV   YUZ_MSN,#00H           ;Bir sonraki dögü için YUZ_MSN'yi
                                   ;sıfırlayarak hazır tut.
```

```
;----

;--------- SANİYE İŞLEMLERİ

      CALL  SANIYE_ISLM            ;Her saniye yapılacak işlem bölümü

      INC   SANIYE                 ;SANIYE değişkenini bir arttır.
      MOV   A,SANIYE               ;Karşılaştırma yapmak için SANIYE'e
                                   ; A'ya aktarılır.
      CJNE  A,#3CH,GERIDON         ;Eğer A (SANIYE) 60'a ulaştıysa
                                   ;devam et,yoksa GERIDON'e atla.
      MOV   SANIYE,#00H            ;Bir sonraki dögü için YUZ_MSN'yi
                                   ; sıfırlayarak hazır tut.
;----

GERIDON:                           ;Timer ISR için bitiş etiketi

      POP   DPL                    ;
      POP   DPH                    ;Program akışında gereklli olan
                                   ; registerların
                                   ;ilk değerleri stack'ten geri çağrılır,
      POP   PSW                    ;programın akışına kaldığı yerden
      POP   B                      ; devam edilir.
      POP   ACC                    ;

RETI                               ;Kesme noktasına geri dön

;*********************************************

;*******************************
SANIYE_ISLM:

      CPL P3.5
;*******************************

INITIALIZE:
      MOV   ON_MSN,#00H
      MOV   YUZ_MSN,#00H
      MOV   SANIYE,#00H

      ORL   PLLCON,#00000011B      ;CORE CLOCK PLL YARDIMIYLA 2.097152
                                   ;MHz clock değeri seçilir.
      MOV   TMOD,#00000001B        ;TİMER-0 MOD-1 SEÇİLDİ.

      MOV   TH0,#0F9H              ;10msn için gerekli olan değer
      MOV   TL0,#02CH              ;TH0 ve TL0'a atanır

      SETB  EA                     ;TÜM INTERRRUPTLAR AÇILDI.
      SETB  ET0                    ;TIMER-0 İNTERRUPT'I AÇILDI.

      SETB  TR0                    ;TIMER-0 ÇALIŞTIRILIR.

      RET
;*************************************************

END
```

Ek Bilgi: Bu uygulamada sadece bir Timer/Counter kullanıldığı için TMOD registeri

```
MOV    TMOD,#00000001
```

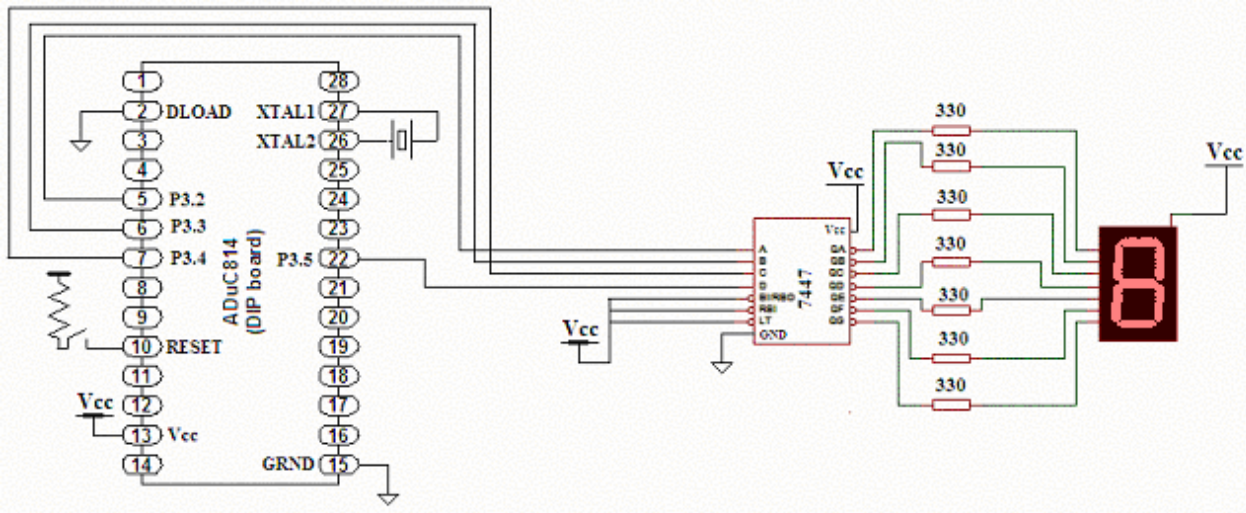
benzeri bir komutla ayarlanmıştır. Bu tarz bir yaklaşım donanım üzerinde çalışan uygulamalarda hiçbir sorun oluşturmayacaktır. Ancak ASPIRE DeBugger gibi IDE'ler kullanarak seri port üzerinden yapılan DeBug çalışması için Timer-1 tabanlı Baud Rate otomatik olarak kullanıldığından bu tür TMOD ayarı MikroConverter-Bilgisayar haberleşmesini kapatabilir. Bunu engellemek için izlenmesi gereken yol maskeleme işlemidir. Örneğin yukardaki uygulamada TMOD ayarlanırken:

```
ANL    TMOD,#11110001    ;1,2,3 nolu bitler "0"seçildi
ORL    TMOD,#00000001    ;0 nolu bit "1" seçildi
                         ;diğer bitler etkilenmedi
```

9.3.9 Tek hane 7-Segment 0-9 Saniye Sayacı

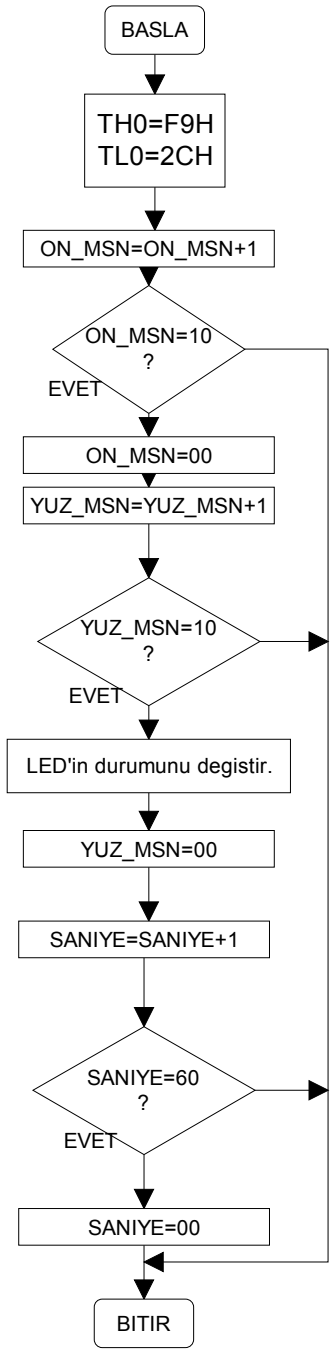
Ön Bilgi:Önceki uygulamalarda hazırladığımız Tek Hane 7-Segment Sürme ve Timer0-ISR program kodlarını birkaç ekle zenginleştirerek basit bir saniye sayacı üretebiliriz. Ek olarak tek yapmamız gereken her saniye için tuttukları değerleri 7447 entegresi için uygun değerlere dönüştüren bir alt rutin kullanmaktır.Bu rutin için bir Look Up Table kullanacağız.

Devre Bağlantıları:

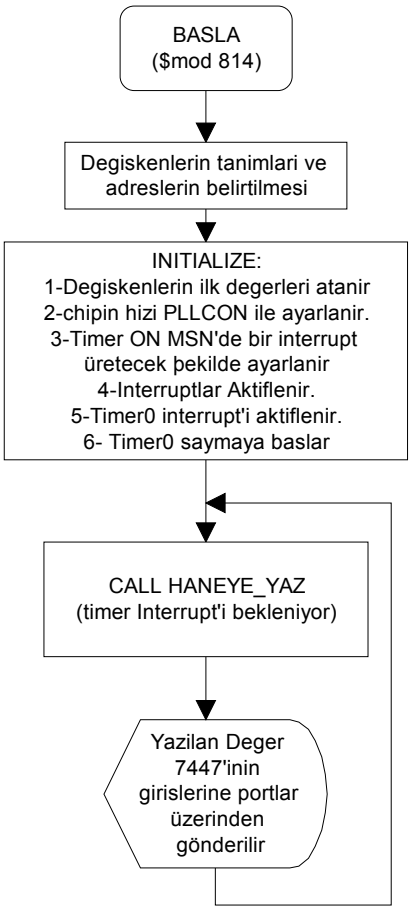


Akış Diagramı:

Timer0 Kesme ServisProgrami



Ana Program



Program Kodu:

```
$MOD814

ON_MSN          DATA  07FH       ;on milisaniyeyi saymak için
                                 ;gerekli sayaç tanımlandı.
```

```
      YUZ_MSN           DATA  07EH        ;yüz milisaniyeyi saymak için
                                          ;gerekli sayaç tanımlandı.
      SANIYE            DATA  07DH        ;saniyeyi saymak için gerekli sayaç
                                          ;tanımlandı.
      SAN_BUF           DATA  07CH        ;Hane sayacı değişkeni

      ORG   0000H                         ;
                                          ;
      JMP   ANA_PROGRAM                   ;
                                          ;Kontrol etmek istediğimiz vektör
      ORG   000BH                         ;adreslerinin girilmesi
                                          ;
      JMP   TMR0_ISR                      ;
                                          ;
      ORG   0060H                         ;

;**************************************
ANA_PROGRAM:

      CALL  INITIALIZE              ;Ön koşulama ve ayarların yapıldığı
                                    ;bölümü çağır.

;--------------------------
ANA_DONGU:                          ;ana programın sonsuz döngüsü

      CALL  HANEYE_YAZ              ;haneyi yazdırma işlemi

      JMP   ANA_DONGU               ;Ana_Dongu basa döner
;--------------------------
;**************************************
TMR0_ISR:


      MOV   TL0,#02CH               ; Döngü içerisinde timer-0'a
      MOV   TH0,#0F9H               ; gerekli değerler yüklenir.


      PUSH  ACC                     ;
      PUSH  B                       ;
      PUSH  PSW                     ;Program akışında gerekli olan
                                    ;registerlar stack'e atılarak
      PUSH  DPH                     ;Saklanır.
      PUSH  DPL                     ;

;-------- ON_MSN İŞLEMLERİ

      INC   ON_MSN                  ;ON_MSN değişkeni bir artırılır.
      MOV   A,ON_MSN                ;Karşılaştırma yapmak için ON_MSN'e
                                    ;A'ya aktarılır.

      CJNE  A,#0AH,GERIDON          ;Eğer A (ON_MSN) 10'a ulaştıysa
                                    ;devam et,yoksa GERIDON'e atla.

      MOV   ON_MSN,#00H             ;Bir sonraki döngü için ON_MSN'yi
                                    ;sıfırlayarak  hazır tut.
;----

;--------- YUZ_MSN İŞLEMLERİ
```

```
      INC   YUZ_MSN                ;YUZ_MSN değişkenini bir arttır.
      MOV   A,YUZ_MSN              ;Karşılaştırma yapmak için
                                   ; YUZ_MSN'e A'ya aktarılır.
      CJNE  A,#0AH,GERIDON         ;Eğer A (YUZ_MSN) 10'a ulaştıysa
                                   ; devam et,yoksa GERIDON'e atla.
      MOV   YUZ_MSN,#00H           ;Bir sonraki dögü için YUZ_MSN'yi
                                   ; sıfırlayarak hazır tut.
;----

;--------- SANİYE İŞLEMLERİ
      INC   SANIYE                 ;SANIYE değişkenini bir arttır.

      CALL  SANIYE_ISLM            ;Her saniye yapılacak işlem bölümü

      MOV   A,SANIYE               ;Karşılaştırma yapmak için SANIYE'e
                                   ; A'ya aktarılır.
      CJNE  A,#3CH,GERIDON         ;Eğer A (SANIYE) 60'a ulaştıysa
                                   ;devam et,yoksa GERIDON'e atla.
      MOV   SANIYE,#00H            ;Bir sonraki dögü için YUZ_MSN'yi
                                   ; sıfırlayarak hazır tut.
;----

GERIDON:                           ;Timer ISR için bitiş etiketi

      POP   DPL                    ;
      POP   DPH                    ;Program akışında gereklli olan
                                   ; registerların
                                   ;ilk değerleri stack'ten geri
                                   ;çağrılır,böylece
      POP   PSW                    ;programın akışına kaldığı yerden
      POP   B                      ;devam edilir.
      POP   ACC                    ;

RETI                               ;Kesme noktasına geri dön

;*******************************
SANIYE_ISLM:

      INC   SAN_BUF                ;Hanede görüntülencek sayı her
      MOV   A, SAN_BUF             ;saniye bir artar
      CJNE  A,#10,DON_SN_ISLM      ;ve 10 olduğunda sıfırlanır
      MOV   SAN_BUF,#0H

DON_SN_ISLM:

      RET
;*******************************

INITIALIZE:
      MOV   SAN_BUF,#00H           ;başlangıçtaki değerler sıfırlanır
      MOV   ON_MSN,#00H
      MOV   YUZ_MSN,#00H
      MOV   SANIYE,#00H


      ORL   PLLCON,#00000011B      ;CORE CLOCK PLL YARDIMIYLA 2.097152
                                   ;MHz clock değeri seçilir.
```

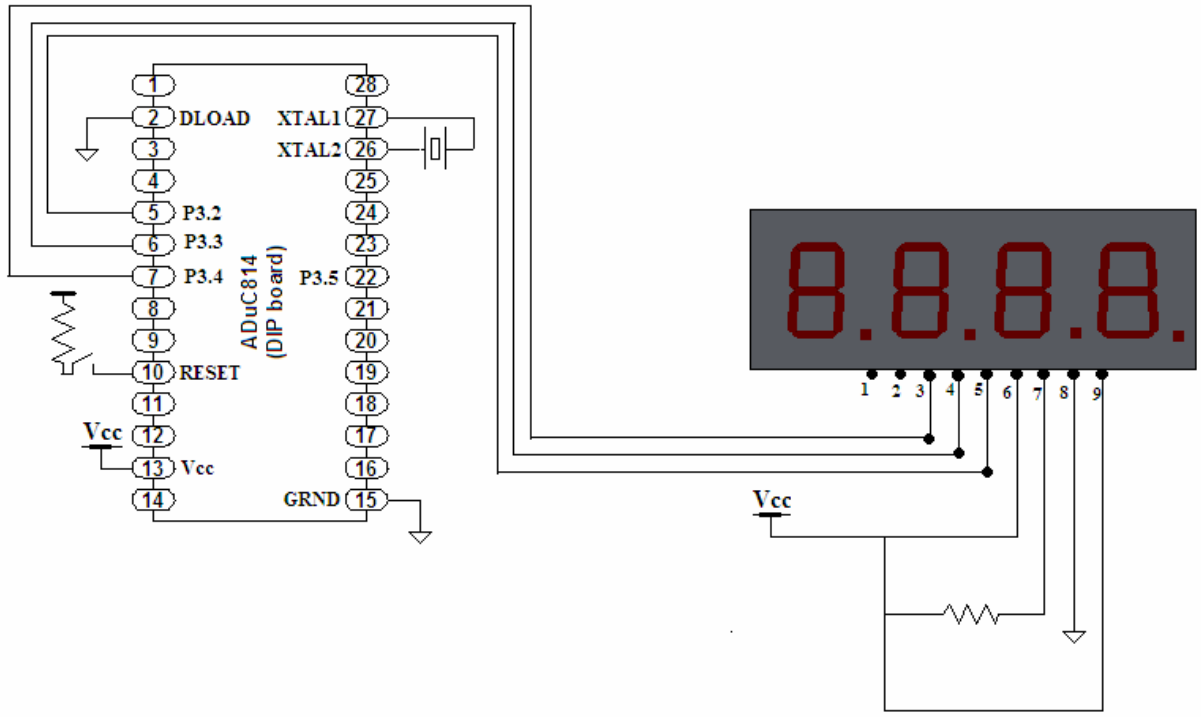
```
      MOV   TMOD,#00000001B         ;TIMER-0 MOD-1 SEÇİLDİ.

      MOV   TH0,#0F9H               ;10msn için gerekli olan değer
      MOV   TL0,#02CH               ;TH0 ve TL0'a atanır

      SETB  EA                      ;TÜM INTERRRUPTLAR AÇILDI.
      SETB  ET0                     ;TIMER-0 İNTERRUPT'I AÇILDI.

      SETB  TR0                     ;TIMER-0 ÇALIŞTIRILIR.

      RET
;************************************************
HANEYE_YAZ:

      MOV   A,SAN_BUF
      MOV   DPTR,#SEV_SEG_TAB           ;look up table'ın başlangıç adresi
                                        ; DPTR'ye atanır
      MOVC  A,@A+DPTR                   ;A ya tablonun istenen değeri
                                        ;atanır

                                        ;ilk bit
      RRC   A
      MOV   P3.2,C
                                        ;ikinci bit
      RRC   A
      MOV   P3.3,C
                                        ;üçüncü bit
      RRC   A
      MOV   P3.4,C
                                        ;dördüncü bit
      RRC   A
      MOV   P3.4,C

      RET


;****************************************
SEV_SEG_TAB:                        ;Hanede görünecek olanı secen
                                    ;tablodur

        DB  00000000B               ;0 Görünür
        DB  00000001B               ;1 Görünür
        DB  00000010B               ;2 Görünür
        DB  00000011B               ;3 Görünür
        DB  00000100B               ;4 Görünür
        DB  00000101B               ;5 Görünür
        DB  00000110B               ;6 Görünür
        DB  00000111B               ;7 Görünür
        DB  00001000B               ;8 Görünür
        DB  00001001B               ;9 Görünür
;************************************************

END
```

Ek Bilgi: Bu uygulama Look Up Table,timer ve her türlü görüntüleme işlemi gerektiren uygulamalarda kullanılacak olan yöntemin en temel halini göstermektedir. Uygulamalar ne kadar değişirse değişsin bu yöntem her durumda verimli olabileceği şekilde ufak modifikasyonlardan sonra kullanılabilir

9.3.10 Intelligent display ile dakika saniye sayıcı uygulaması

Ön Bilgi: Bu uygulamada bir önceki uygulamada olduğu gibi bir saniye sayacı oluşturacağız. Sadec görüntülenecek olan zaman bu kez 60 dakikaya kadar çıkabilecek ve 4 haneli bir Intelligent Display ile görüntülenecektir.

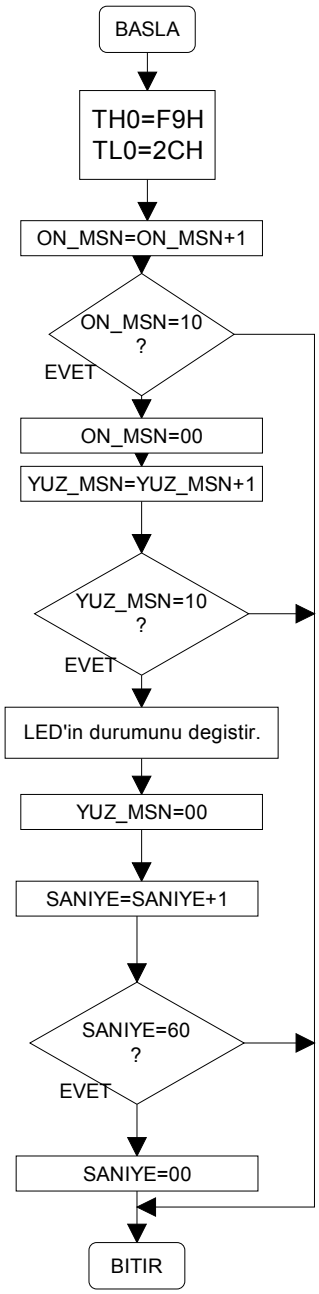
Devre Bağlantıları:



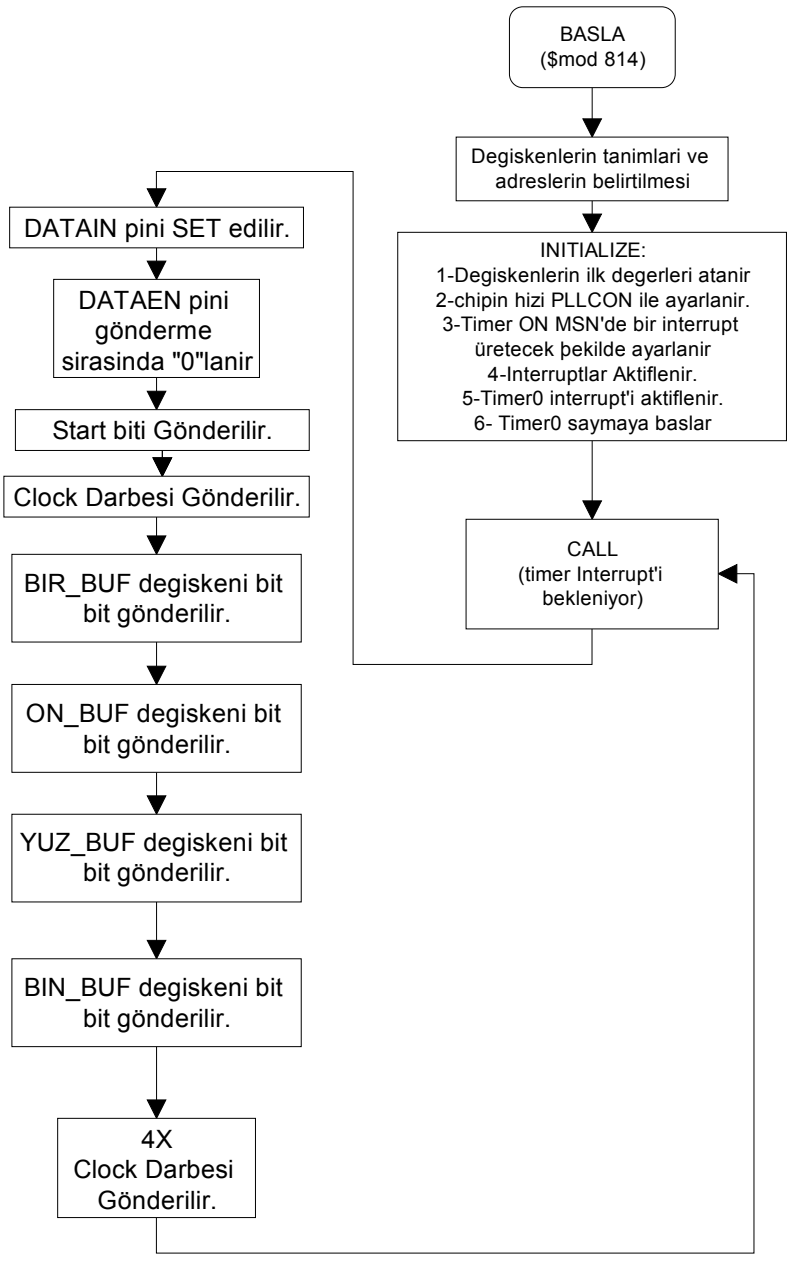
Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Akış Diagramı:

Timer0 Kesme ServisProgrami



Ana Program



Program Kodu:

```
$MOD814
```

```
      ON_MSN            DATA  07FH        ;on msn sayacı tanımlandı
      YUZ_MSN           DATA  07EH        ;yüz msn sayacı tanımlandı
      SANIYE            DATA  07DH        ;Saniye sayacı tanımlandı
      SAN_BIR           DATA  07CH        ;Saniyenin birler basamağını
                                          saklamak için kullanılan buffer.
      SAN_ON            DATA  07BH        ;Saniyenin onlar basamağını
                                          saklamak için kullanılan buffer.
      DAK_BIR           DATA  07AH        ;Dakikanın birler basamağını
                                          ;saklamak için kullanılan buffer.
      DAK_ON            DATA  079H        ;Dakikanın onlar basamağını
                                          ;saklamak için kullanılan buffer.
      SAN_BIR_BUF       DATA  078H        ;Saniyenin birler basamağının
                                          ;düzenlenmiş halini saklamak için
                                          ;kullanılan buffer
      SAN_ON_BUF        DATA  077H        ;Saniyenin onlar basamağının
                                          ;düzenlenmiş halini saklamak için
                                          ;kullanılan buffer.
      DAK_BIR_BUF       DATA  076H        ;dakikanın birler basamağının
                                          ;düzenlenmiş halini saklamak için
                                          ;kullanılan buffer.
      DAK_ON_BUF        DATA  075H        ;Dakikanın onlar basamağının
                                          ;düzenlenmiş halini saklamak için
                                          ;kullanılan buffer.

      SAYAC             DATA  074H        ;Çok amaclı sayac

      CLOCK             EQU   P3.2        ;Clock sinyalinin çıkış pini.
      DATAIN            EQU   P3.3        ;DATAIN pini.
      DATAEN            EQU   P3.4        ;Bilgi giriş pini.

      ORG   0000H                         ;
                                          ;
      JMP   ANA_PROG                      ;
                                          ;Kontrol etmek istediğimiz vektör
      ORG   000BH                         ; adreslerinin girilmesi
                                          ;
      JMP   TMR0_ISR                      ;
                                          ;
      ORG   0060H                         ;


ANA_PROG:
      CALL  INITIALIZE                    ;Ön hazırlıkların yapıldığı
                                          ;altprogramının çağrılması.
ANA_DONGU:

      CALL  DISPLAY_SUR                   ;İstenilen değerlerin Displayde
                                          ;görüntülenmesini sağlayan
                                          ;çağrılması altprogramın

      JMP   ANA_DONGU                     ;ANA PROGRAM DÖNGÜSÜNÜN SONU

;**************************************
TMR0_ISR:


     MOV    TL0,#02CH                    ; Döngü içerisinde timer-0'a
```

```
      MOV   TH0,#0F9H                  ; gerekli değerler yüklenir.


      PUSH  ACC                        ;
      PUSH  B                          ;
      PUSH  PSW                        ;Program akışında gerekli olan
                                       ;registerlar stack'e atılarak
      PUSH  DPH                        ;Saklanır.
      PUSH  DPL                        ;
;-------- ON_MSN İŞLEMLERİ

      INC   ON_MSN                     ;ON_MSN değişkeni bir artırılır.
      MOV   A,ON_MSN                   ;Karşılaştırma yapmak için ON_MSN'e
                                       ;A'ya aktarılır.

      CJNE  A,#0AH,GERIDON             ;Eğer A (ON_MSN) 10'a ulaştıysa
                                       ;devam et,yoksa GERIDON'e atla.

      MOV   ON_MSN,#00H                ;Bir sonraki döngü için ON_MSN'yi
                                       ;sıfırlayarak  hazır tut.
;----

;--------- YUZ_MSN İŞLEMLERİ

      INC   YUZ_MSN                    ;YUZ_MSN değişkenini bir arttır.
      MOV   A,YUZ_MSN                  ;Karşılaştırma yapmak için
                                       ;YUZ_MSN'e A'ya aktarılır.
      CJNE  A,#0AH,GERIDON             ;Eğer A (YUZ_MSN) 10'a ulaştıysa
                                       ; devam et,yoksa GERIDON'e atla.
      MOV   YUZ_MSN,#00H               ;Bir sonraki dögü için YUZ_MSN'yi
                                       ;sıfırlayarak hazır tut.
;----

;--------- SANİYE İŞLEMLERİ
      INC   SANIYE                     ;SANIYE değişkenini bir arttır.

      CALL  SANIYE_ISLM                ;Her saniye yapılacak işlem bölümü

      MOV   A,SANIYE                   ;Karşılaştırma yapmak için SANIYE'e
                                       ;A'ya aktarılır.
      CJNE  A,#3CH,GERIDON             ;Eğer A (SANIYE) 60'a ulaştıysa
                                       ;devam et,yoksa GERIDON'e atla.
      MOV   SANIYE,#00H                ;Bir sonraki dögü için YUZ_MSN'yi
                                       ; sıfırlayarak hazır tut.
;----

GERIDON:                               ;Timer ISR için bitiş etiketi

      POP   DPL                        ;
      POP   DPH                        ;Program akışında gereklli olan
                                       ; registerların
                                       ;ilk değerleri stack'ten geri
                                       ;çağrılır,böylece
      POP   PSW                        ;programın akışına kaldığı yerden
      POP   B                          ;devam edilir.
      POP   ACC                        ;

RETI                                   ;Kesme noktasına geri dön
```

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

```
;*******************************
SANIYE_ISLM:

      CALL  DISPLAY_HAZIRLA

      RET
;*******************************
DISPLAY_HAZIRLA:

      INC   SAN_BIR                    ;her saniyede SAN_BIR'i bir
                                       ;arttırır.
      MOV   A,SAN_BIR                  ;karşılaştırma için SAN_BIR'i
                                       ;ACC'ye yükle.

      CJNE  A,#10,DON_DISPLAY_HAZIRLA  ;SAN_BIR 10 olana kadar
                                       ; bekle ve
                                       ;sonra devam et.
      MOV   SAN_BIR,#0H                ;Eğer SAN_BIR 10'a
                                       ;ulaşmışsa sıfırla

SAN_ON_DOLDU:
      INC   SAN_ON                     ;her 10 saniyede bir SAN_ON'u
                                       ; bir arttır.
      MOV   A,SAN_ON                   ;karşılaştırma için SAN_ON'u ACC'ye
                                       ;kopyala.
      CJNE  A,#6,DON_DISPLAY_HAZIRLA   ;SAN_ON 6 olana kadar
                                       ; bekle ve
                                       ;sonra devam et.
      MOV   SAN_ON,#0H                 ;Eğer SAN_ON 6'ya
                                       ;ulaşmışsa sıfırla

DAK_BIR_DOLDU:
      INC   DAK_BIR                    ;her bir dakika sonunda bu buffer'ı
                                       ;bir arttırır.
      MOV   A,DAK_BIR                  ;karşılaştırma için DAK_BIR'u
                                       ; ACC'ye kopyala.


      CJNE  A,#10,DON_DISPLAY_HAZIRLA  ;DAK_BIR 10 olana kadar
                                       ; bekle ve
                                       ;sonra devam et.
      MOV   DAK_BIR,#0H                ;Eğer DAK_BIR 10'a
                                       ;ulaşmışsa sıfırla

DAK_ON_DOLDU:
      INC   DAK_ON                     ;her on dakika sonunda bu buffer'ı
                                       ;bir arttırır.
      MOV   A,DAK_ON                   ;karşılaştırma için DAK_ON'u ACC'ye
                                       ; kopyala.

      CJNE  A,#6,DON_DISPLAY_HAZIRLA   ;DAK_ON 6 olana kadar
                                       ; bekle ve
                                       ;sonra devam et.
      MOV   DAK_ON,#0H                 ;Eğer DAK_ON 6'ya
                                       ;ulaşmışsa sıfırla

DON_DISPLAY_HAZIRLA:
```

```
      CALL  DONUSTURUCU                   ;Sayıları displaye göndermeden
                                          ;önce gerekli koda dönüştüeren
                                          ;altprogramın çağrılması.
      RET
;*********************************************
CLOCK_GETIR:
      SETB  CLOCK                         ;Gerekli olan Clock sinyalinin
      CLR   CLOCK                         ;verilmesi.
      RET

;*********************************************
DISPLAY_SUR:

      CLR   DATAEN                        ;Bilgi aktarımına başlamak için
                                          ;DATAEN pini "0"a çekilir.
      SETB  DATAIN                        ;DATAIN pini "1" yaparak "start
                                          ;biti"nin gönderilmesi

CALL  CLOCK_GETIR                         ;Bir clock darbesi gönder.

                                          ;dakikanın ikinci basamaığını gönder

      MOV   A,DAK_ON_BUF                  ;
      CALL BIT_BIT_GONDER                 ;DAK_ON_BUF'ın ekrana yazılması
      CLR   DATAIN                        ;için gerekli komutlar.
      CALL  CLOCK_GETIR                   ;

      ;dakikanın BİRİNCİ basamaığını gönder,

      MOV   A,DAK_BIR_BUF                 ;
      CALL BIT_BIT_GONDER                 ;DAK_BIR_BUF'ın ekrana yazılması
      CLR   DATAIN                        ;için gereken komutlar.
      CALL  CLOCK_GETIR                   ;

                                          ;SANİYE ikinci basamaığını gönder
      MOV   A,SAN_ON_BUF                  ;
      CALL BIT_BIT_GONDER                 ;SAN_ON_BUF'ın ekrana yazılması
      CLR   DATAIN                        ;için gereken Komutlar
      CALL  CLOCK_GETIR                   ;

                                          ;saniye birinci basamaığını gönder
      MOV   A,SAN_BIR_BUF                 ;
      CALL BIT_BIT_GONDER                 ;SAN_BIR_BUF'ı ekrana yazılması
      CLR   DATAIN                        ;için gereken komutlar.
      CALL  CLOCK_GETIR                   ;

      CLR   DATAIN                        ;
      CALL  CLOCK_GETIR                   ;İnt.Disp. için gerekli 36 bit
      CLR   DATAIN                        ;uzunluktaki data'yı tamamlamak
      CALL  CLOCK_GETIR                   ;için kullanılır.
      CALL  CLOCK_GETIR                   ;
      CALL  CLOCK_GETIR                   ;
      SETB  DATAEN                        ;DATAEN pini "1"a çekilir.İnt.
                                          ;disp'e bilgi girişi durur.
      RET
;*********************************************

DONUSTURUCU:
```

```
                                        ;Saniye birler basamağını dönüştür
      MOV   A,SAN_BIR
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   SAN_BIR_BUF,A

                                        ;Saniye onlar basamağını dönüştür
      MOV   A,SAN_ON
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   SAN_ON_BUF,A

                                        ;dakika birler basamağını dönüştür
      MOV   A,DAK_BIR
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   DAK_BIR_BUF,A

                                        ;dakika onlar basamağını dönüştür
      MOV   A,DAK_ON
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   DAK_ON_BUF,A
      RET

;*********************************************

BIT_BIT_GONDER:
      MOV   SAYAC,#07                   ;dögü için gerekli sayı atanır.
TUR:
      RRC   A                           ;ACC C üzerinde rotate edilir.
      MOV   DATAIN,C                    ;C ile bit bit gönderim yapılır.
      CALL  CLOCK_GETIR                 ;Clock sinyali verilir.

      DJNZ  SAYAC,TUR                   ;7-bit gönderilene kadar bu
                                        ;işlemler devam eder.
      RET

;*********************************************

INITIALIZE:
      MOV   SAN_BIR,#00H                ;
      MOV   SAN_ON,#00H                 ;
      MOV   DAK_BIR,#00H                ;Kullanılan Buffer'ların
      MOV   DAK_ON,#00H                 ;ilk koşulları atandı.
      MOV   SAN_BIR_BUF,#00H
      MOV   SAN_ON_BUF,#00H
      MOV   DAK_BIR_BUF,#00H
      MOV   DAK_ON_BUF,#00H
      MOV   SAYAC,#00H

      CLR   CLOCK                       ;
      CLR   DATAIN                      ;Kullanılan port bacaklarının
      CLR   DATAEN                      ;ilk koşulları atandı


      ORL   PLLCON,#00000011B           ;CORE CLOCK_GETIR PLL YARDIMIYLA
                                        ;2.097152 MHz'E AYARLANDI.
```

```
      MOV   TMOD,#00000001B               ;TİMER-0 MOD-1 SEÇİLDİ.

      MOV   TH0,#0F9H                     ;10mili  saniye  için  gerekli  olan
değer
      MOV   TL0,#02CH                     ;TH0 ve TL0'a atanır
      SETB  EA                            ;TÜM INTERRRUPTLAR AÇILDI.
      SETB  ET0                           ;TIMER-0 İNTERRUPT'I AÇILDI.

      SETB  TR0                           ;TIMER-0 ÇALIŞTIRILIR.

      RET
;*************************************************



SEV_SEG_TAB:                              ;Hanede görünecek olanı secen
                                          ; tablodur

        DB  00111111B                     ;0 Görünür
        DB  00000110B                     ;1 Görünür
        DB  01011011B                     ;2 Görünür
        DB  01001111B                     ;3 Görünür
        DB  01100110B                     ;4 Görünür
        DB  01101101B                     ;5 Görünür
        DB  01111101B                     ;6 Görünür
        DB  00000111B                     ;7 Görünür
        DB  01111111B                     ;8 Görünür
        DB  01101111B                     ;9 Görünür
;*************************************************

END
```

Ek Bilgi: Son iki uygulamada görüldüğü gibi bir kez yazılan verimli bir Timer ISR'si farklı uygulamalarda aynı şekilde kullanılabilir.Bu yüzden kendi görevini yerine getiren ve bu görev dışında hiçbir olaya etki etmeyen alt rutinlerin saklanması ve bu tip program parçalarından oluşan bir kütüphanenin kurulması verimli ve hızlı proje üretimi açısından önemlidir.

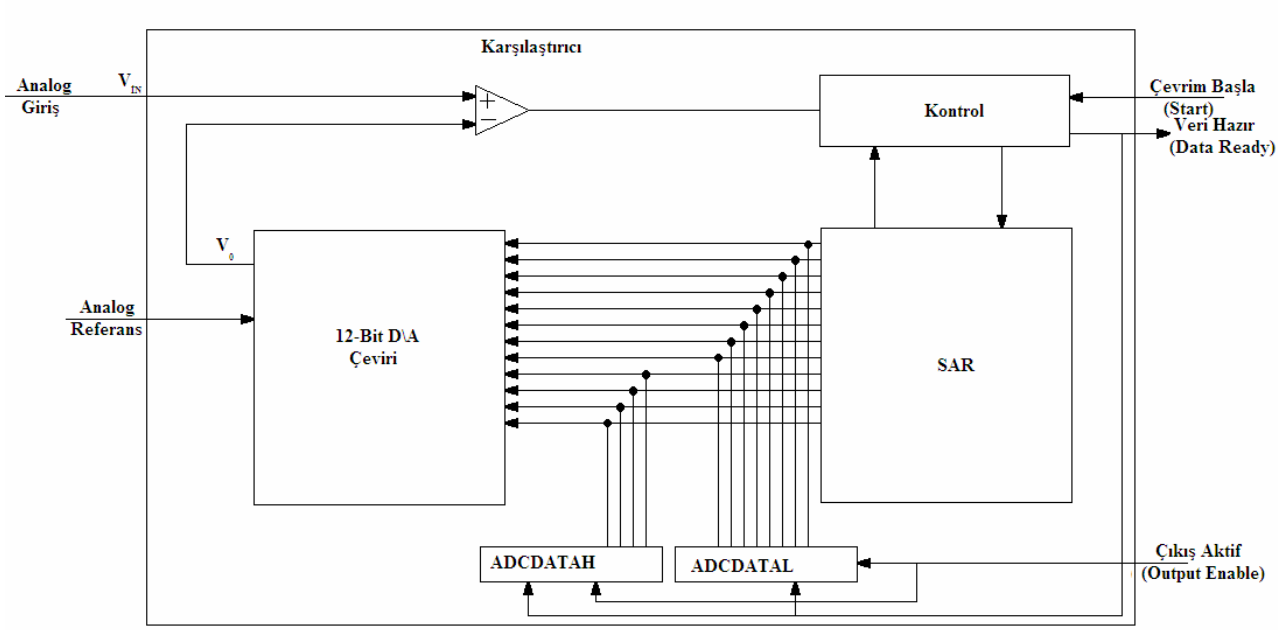
9.3.11 8-BİT ADC BİLGİSİNİN OKUNMASI VE GÖRÜNTÜLENMESİ

Ön Bilgi: Bu uygulama ile ADuC814 microconverter' ın gerçek kullanım amacı olan ADC işlemine göz atacağız.Aşağıda devre şemasında da gördüğünüz gibi bir gerilim bölücüyle ikiye ayırdığımız 5V V_{cc} değerini 0-2.5V arasındaki değerleri elde etmek için bir ayarlı direnç yardımıyla tekrar böldük. Burada 2.5V maksimum değerde işlem yapmamızın nedeni ADuC serisi ile 0-2.5V aralığında ADC okunabilmesidir. Bu konu 8.2.2 A/D Dönüşüm bölmünde daha detaylı olarak incelenmiştir.Bu uygulamada okuduğumuz 12-bit ADC değerinin en düşük 4-bit' ini atarak 8-bitlik gerilim değerini intelligent 7-segment display kullanarak 3 hane şeklinde görüntüleyeceğiz.Ancak uygulamadan önce ADC işlemini kontrol eden ve gerekli sayıları tutan registerlar hakkında yeterli bilgiye ihtiyaç vardır.

Öncelikle bilinmesi gereken ADuC814 Microconverter SAR (Succesive Aproximation) Yani Gerçek Değer Yaklaştırmalı ADC işlemi kullanan bir yapıya sahiptir.

Gerçek Değer Yaklaştırma A\D çevrim tekniği DAC çıkışı olan V0 ile Analog giriş sinyali Vin'in karşılaştırılmasına dayanır.DAC'ın sayısal girişleri gerçek Değer Yaklaştırma Metodu ile üretilir.DAC çıkışı anaşog sinyaline eşit olduğunda DAC'ın giriş analog sinyalin sayısal değerini verir. Bu değer çıkış register'ı (ADCDATAH ve ADCDATAL) ile dışarıya alınır.

Gerçek Değer Yaklaştırma Metodu: DAC'a giriş üreten bu teknik ağırlığı bilinmeyen bir maddenin (Örneğin; 1gr'dan daha az) 1 /2 gr ,1/4 gr,1/8 gr… gibi kesirli agırlıklarla bir terazide tartılmasına benzer. Tarma işlemi en ağır (1/2 gr.) ile başlar ve denge bozulana kadar daha düşük ağırlıklar(azalan sırada) eklenir.Denngeyi bozan ağırlık çıkartılır ve işlem en küçük ağırlık kullanılıncaya kadar devam eder.



Yukarıdaki şekilde 12-bit ADC'ye anlatılan gerçek değer yaklaştırma metodunu uygularsak öncelikle en yüksek değerlikli bit (ADCDATAH'ın 3.biti) set edilir ve değer gerçek değer ile karşılaştırılır.Eğer karşılaştırıcı durumu değiştirirse bu yüksek değerlikli bit tarafında üretilen çıkışın Analog sinyalden büyük olduğunu gösterir.SAR (Successive Approximation Register) 'daki en yüksek bit lojik "0" bir sonraki daha düşük değerlikli bit ise aktif edilir ve aynı işlemler en düşük değerlikli bite kadar tekrarlandığında ADC'de okunan sinyale yaklaşık bir değer elde edilir.

ADuC814'te ADC uygulaması yapmak için ADC kontrolü ile ilgili ADCCON1, ADCCON2 ve ADCCON3 registerlarının ayarlanması gerekmektedir.

ADCCON1 Register'ı

7	6	5	4	3	2	1	0
MODE	EXT_REF:	CK1	CK0	AQ1	AQ0	T2C	EXC

MODE: ADC'yi çalıştırmaya yarayan bittir. Lojik "1" yapıldığında ADC kullanım için açılmış olmaktadır.

EXT_REF: Harici referans seçme bitidir.Lojik"1" yapıldığında harici referans pini, lojik"0" yapıldığında on-chip referansı kullanılır.On-Chip referans Maximum 2.5V'tur.

CK1 :ADC clock oluşturmak için kullanılan PLL master Clock için bölüm oranı
CK0 : kullanılırlar seçmek amacı ile
AQ1 :Giriş sinyalini ortalamasını almak için kullanılan giriş track\hold amplifier için
AQ0 : gerekli olan ADC clock'larının sayısını seçmek amacıyla kullanılırlar
T2C :Timer-2 taşma bitini ADC Dönüşümünü tetikleyen giriş
EXC :Bir kenat tetiklemesi vasıtasıyla ADC işleminin başlatmaktadır:Set edildiğinde gelen kenar tetiklemesi ile ADC işlemini başlatır.

ADCCON2 Register'ı

7	6	5	4	3	2	1	0
ADCI	ADCSPI	CCONV	SCONV	CS3	CS2	CS1	CS0

ADCI : ADC interrupt bayrağıdır. Donanım tarafından birlenir.Lojik "1" olduğu durumda ADC işleminin bittiğini gösterir.
ADCSPI : bu bitin set edilmesiyle ADC işleminin sonucu herhangi bir işlem yapılmadan SPI data buffer'ının (SPIDAT) içerisine otomatik olarak yüklenir.
CCONV : Bu bitin set edilmesiyle ADC bir ADC işlemini tamamladıktan sonra durmadan ikinci ADC işlemine başlayacaktır.
SCONV : Bu bit set edildiğinde ADC bir adet ADC işlemi yapıp duracak ve başka bir komut gelene kadar ADC işlemi yapmayacaktır.
CS3,CS2,CS1, CS0: Bu bitler yardımıyla istenilen kanal seçimi yapılmaktadır.

CS3	CS2	CS1	CS0	Kanal
0	0	0	0	ADC0
0	0	0	1	ADC1
0	0	1	0	ADC3
0	0	1	1	ADC4
0	1	0	0	ADC5
0	1	0	1	ADC kanal seçimi yok.
0	1	1	0	ADC kanal seçimi yok
1	0	0	0	On-Chip sıcaklık sensörü
1	0	0	1	DAC0
1	0	1	0	DAC1
1	0	1	1	AGND
1	1	0	0	VREF

ADCCON3 Register'ı

BUSY	GNCLD	AVGS1	AVGS0	OFCLD	MODCAL	TYPECAL	SCAL

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

BUSY : sadece okunabilen bir bittir bu bitin "1" olduğu durumda ADC işlemi yapılıyor, anlamındadır.
GNCLD :Gain kalibrasyon kapatma biti (0=Açık)
AVGS1 ve AVGS0: Bu iki bitin seçimi ile ADC işleminin sonucunun istenildiği kadar ortalamasının alınmasını sağlar.

AVGS1	AVGS0	Ortlamaya alınan değer sayısı
0	0	15
0	1	1
1	0	31
1	1	63

OFCLD :Ofset kalibrasyon kapatma biti (0=Açık)
MODCAL : ADC uygulamalarında kalibrasyon çevrimlerinde bu bit set edilmelidir.
TYPECAL : Bu bit ADC kalibrasyonu sırasında "Offset" mi yoksa "Gain" ayarının mı yapıldığı bildirmektedir.Eğer bu bit lojik "1" yapılırsa "Gain" kalibrasyonu, eğer lojik "0" yapılırsa "Offset" kalibrasyonu yapılmış olur.
SCAL :Bu bit ADC işleminin başlaması sağlamaktadır. Bu bit lojik "1" yapıldığında ADC dönüşüm işlemine başlar, ve bu işlem bittiğinde donanım tarafından lojik "0"a çekilir.

Öncelikle ADC için ADCCON1 , ADCCON2 ve ADCCON3 register'larında bize gerekli şekilde ayarlaması gerekmektedir. Bundan sonra yapılacak işlem ise ADC için "Gain" ve "Offset" kalibrasyonlarının yapılması gerekmektir.Bu ön ayarlamadaki kalibrasyonlar ,ADCCON3'deki TYPECAL bitinin önce "Offset" ve daha sonra "Gain" modlarına ayarladıktan sonra birer kez ADC işlemin yapılması yeterli olacaktır.Örnekte olduğu gibi;

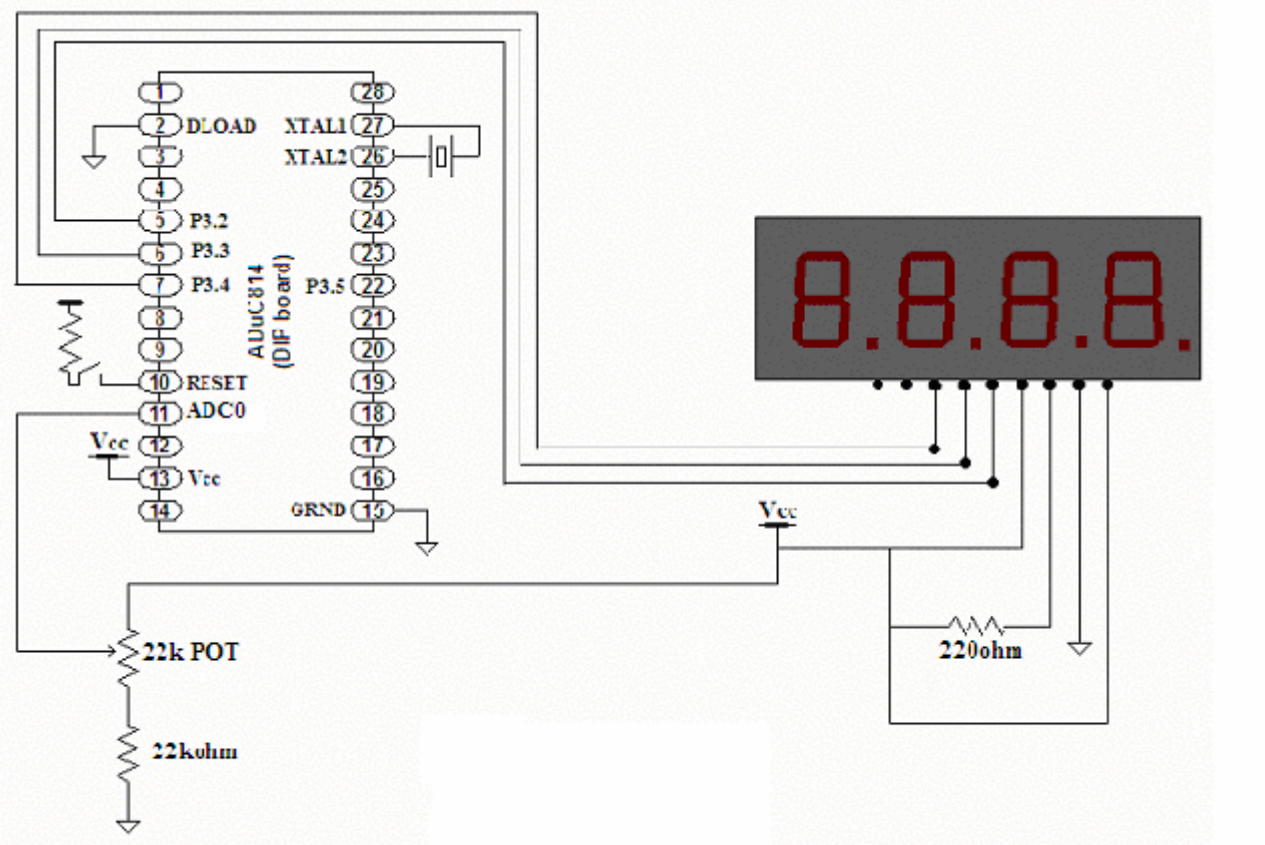
```
CLR   ADCI                  ;ADC bayrağının temizlenmesi.
MOV   ADCCON3,#00000001B    ;ADC'nin ilk kalibbrasyon için offset
                            ;ayarının yapılması için ön
                            ;hazırlık(ikinci bitin "0" yapılması ile
                            ;OFFSET ayarı yapılır)
JNB   ADCI,$                ;ADC kalibrasyon işleminin bitmesini
                            ; bekle
CLR   ADCI                  ;ADC bayrağının temizlenmesi.
MOV   ADCCON3,#00000011B    ;ADC'nin ilk kalibrasyonu için GAİN
                            ;ayarının yapılması için ön hazırlık
                            (İkinici ;bitin "1" yapılması ile GAİN
                            ayarı yapılır.)
JNB   ADCI,$                ;ADC kalibrasyon işleminin bitmesini
                            ;bekle
CLR   ADCI                  ;ADC bayrağının temizlenmesi.
```

Bu işlemler gerçekleştirildikten sonra ADC dönüşüm işlemi için hazır hale gelmiş olur.ADC işlemini ADCI bayrağını kontrol ederek kontrol edebiliriz. Bundan sonra öncelikle ADCI bayrağını temizleyerek işlemlerimize devam edebiliriz. Bundan sonra ADC işleminin sürekli mi yoksa tek bir kez mi dönüşüm yapacağını SCONV ve CCONV bitlerinden gerekli olanı "1" yaparak seçim yapılır. Bu noktada ADC işlemini başlatmak için ADCCON3'un 0. biti SCAL biti ,set edilerek ADC işlemi başlamış olur.

ADC işleminin bitmesi ile ADCI bayrağı "1" olur. Bu bayrağın kontrolüyle ADC işleminin sonuçlarını bizim tanımladığımız buffer'lar içerisine alabiliriz.Bu alınan bilgileri gerekli işlemlerden geçirdikten sonra yerde kullanabiliriz.Herhangi bir LCD ve 7-segment gibi ekranlara yazdırmak, DAC yardımıyla analog bir sinyal oluşturup ADC'den okunan değere bağlı bir cihaz sürülebilir.

Kullandığımız ADuC 814 çipinin 2 bitlik bir hata oranı vardır.Bu hatadan dolayı 12 bitlik ADC'nin son iki bitini işleme almaz isek ,10-bitlik net bir çözünürlük elde etmiş oluruz.Chipimizin maximum 2.5V'luk bir değere ADC işlemini uygulayabildiğini biliyoruz. Buradan ADC dönüşümünün yaklaşık 2.5mV'luk ($2.5V/2^{10}=2.5/1024=0.0024414V$) bir hassasiyet ile ölçüm yaptığını hesaplayabiliriz.

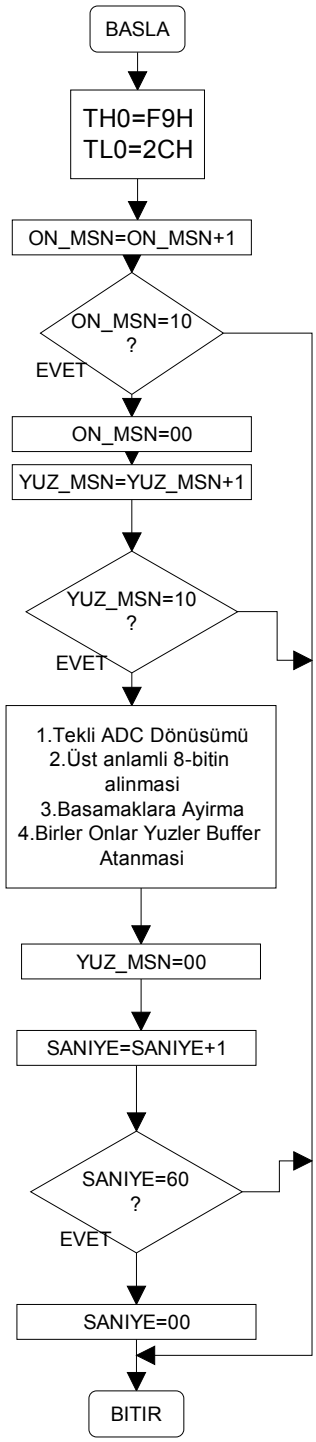
Devre Bağlantıları :

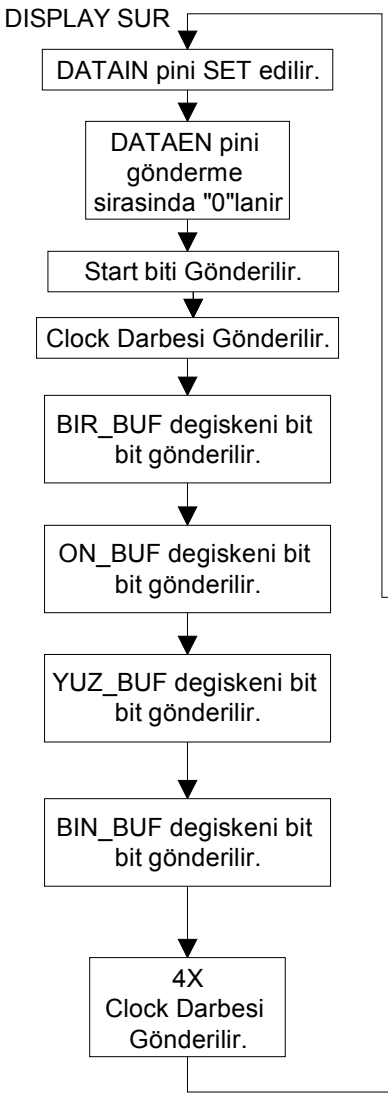


Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

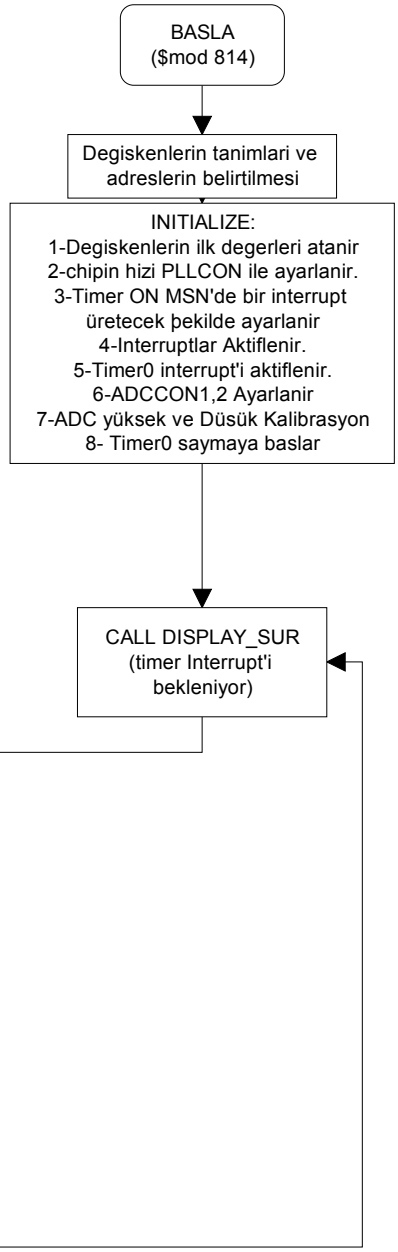
Akış Diagramı:

Timer0 Kesme ServisProgrami





Ana Program



Program Kodu:

```
      $MOD814
                                    ;timer0 bufferları
      ON_MSN            DATA  06DH  ;on msn saymak için gerekli sayaç
      YUZ_MSN           DATA  06CH  ;yüz msn saymak için gerekli sayaç
      SANİYE            DATA  07AH

                                    ;intelligent display bufferları

      BIR_BUF           DATA  079H
      ON_BUF            DATA  078H
      YUZ_BUF           DATA  077H
      VOLT              DATA  076H
      SAYAC             DATA  06AH
      CLOCK             EQU   P3.2
      DATAIN            EQU   P3.3
      DATAEN            EQU   P3.4

                                    ;ADC bufferları
      ADC_SAYAC   DATA  06BH
      ADC0_DEGERI DATA  069H
;*******************************************

      ORG   0000H                   ;
                                    ;
      JMP   ANA_PROG                ;
                                    ;Kontrol etmek istediğimiz vektör
      ORG   000BH                   ; adreslerinin girilmesi
                                    ;
      JMP   TMR0_ISR                ;
                                    ;
      ORG   0060H                   ;

ANA_PROG:
      CALL  INITIALIZE              ;Ön hazırlıkların yapıldığı
                                    ;altprogramının çağrılması.
ANA_DONGU:

      CALL  DISPLAY_SUR             ;İstenilen değerlerin Displayde
                                    ;görüntülenmesini sağlayan
                                    ;çağrılması altprogramın

      JMP   ANA_DONGU               ;ANA PROGRAM DÖNGÜSÜNÜN SONU


;*************************************
TMR0_ISR:

      MOV   TL0,#02CH               ; Döngü içerisinde timer-0'a
      MOV   TH0,#0F9H               ; gerekli değerler yüklenir.


      PUSH  ACC                     ;
      PUSH  B                       ;
      PUSH  PSW                     ;Program akışında gerekli olan
                                    ;registerlar stack'e atılarak
      PUSH  DPH                     ;Saklanır.
```

```
      PUSH  DPL                     ;


;-------- ON_MSN İŞLEMLERİ

      INC   ON_MSN                  ;ON_MSN değişkeni bir artırılır.
      MOV   A,ON_MSN                ;Karşılaştırma yapmak için ON_MSN'e
                                    ;A'ya aktarılır.

      CJNE  A,#0AH,GERIDON          ;Eğer A (ON_MSN) 10'a ulaştıysa
                                    ;devam et,yoksa GERIDON'e atla.

      MOV   ON_MSN,#00H             ;Bir sonraki döngü için ON_MSN'yi
                                    ;sıfırlayarak  hazır tut.
;----

;--------- YUZ_MSN İŞLEMLERİ

      INC   YUZ_MSN                 ;YUZ_MSN değişkenini bir arttır.
      MOV   A,YUZ_MSN               ;Karşılaştırma yapmak için
                                    ;YUZ_MSN'e A'ya aktarılır.
      CJNE  A,#0AH,GERIDON          ;Eğer A (YUZ_MSN) 10'a ulaştıysa
                                    ;devam et,yoksa GERIDON'e atla.
      MOV   YUZ_MSN,#00H            ;Bir sonraki dögü için YUZ_MSN'yi
                                    ; sıfırlayarak hazır tut.
;----

;--------- SANİYE İŞLEMLERİ
      INC   SANIYE                  ;SANIYE değişkenini bir arttır.

      CALL  SANIYE_ISLM             ;Her saniye yapılacak işlem bölümü

      MOV   A,SANIYE                ;Karşılaştırma yapmak için SANIYE'e
                                    ;A'ya aktarılır.
      CJNE  A,#3CH,GERIDON          ;Eğer A (SANIYE) 60'a ulaştıysa
                                    ;devam et,yoksa GERIDON'e atla.
      MOV   SANIYE,#00H             ;Bir sonraki dögü için YUZ_MSN'yi
                                    ;sıfırlayarak hazır tut.
;----

GERIDON:                            ;Timer ISR için bitiş etiketi

      POP   DPL                     ;
      POP   DPH                     ;Program akışında gereklli olan
                                    ; registerların
                                    ;ilk değerleri stack'ten geri
                                    ;çağrılır,böylece
      POP   PSW                     ;programın akışına kaldığı yerden
      POP   B                       ;devam edilir.
      POP   ACC                     ;

RETI                                ;Kesme noktasına geri dön
;*********************************
SANIYE_ISLM:

      CALL  ADC_OKU
      CALL  BASAMAKLARA_AYIRMA
      CALL  DONUSTURUCU
```

```
RET

;*********************************
ADC_OKU:

      CLR   ADCI
      SETB  SCONV
      JNB   ADCI,$

TEK_BYTE_ELDESI:
      MOV   A,ADCDATAH
      ANL   A,#00001111B
      MOV   R0,A
      MOV   A,#00H
      MOV   A,ADCDATAL
      ANL   A,#11110000B
      ADD   A,R0
      SWAP  A
      MOV   ADC0_DEGERI,A
      RET

;*********************************************
BASAMAKLARA_AYIRMA:

      MOV   A,ADC0_DEGERI
      MOV   B,#100
      DIV   AB
      MOV   YUZ_BUF,A         ;Sonuç olarak gelen yüzler basamağını yaz
      MOV   A,B               ;kalanı A ya yaz
      MOV   B,#10             ;B ye 10 ata
      DIV   AB                ;A yı B ye böl
      MOV   ON_BUF,A          ;Onlar basamağını yaz
      MOV   BIR_BUF,B         ;Birler basamağını yaz

      RET
;*******************************
DONUSTURUCU:
                              ;birler basamağını dönüştür
      MOV   A,YUZ_BUF
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   YUZ_BUF,A
                              ;onlar basamağını dönüştür
      MOV   A,ON_BUF
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   ON_BUF,A
                              ;yüzler basamağını dönüştür
      MOV   A,BIR_BUF
      MOV   DPTR,#SEV_SEG_TAB
      MOVC  A,@A+DPTR
      MOV   BIR_BUF,A
      RET
;*********************************************
DISPLAY_SUR:

      CLR   DATAEN
      SETB  DATAIN
```

```
      CALL  CLOCK_GETIR

                              ;sabit sıfır sayısını gönder
      MOV   A,VOLT
      CALL BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

                              ;birler basamağını gönder
      MOV   A,YUZ_BUF
      CALL BIT_BIT_GONDER
      SETB  DATAIN
      CALL  CLOCK_GETIR

                              ;onlar basamağını gönder
      MOV   A,ON_BUF
      CALL BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

                              ;yüzler basamağını gönder
      MOV   A,BIR_BUF
      CALL BIT_BIT_GONDER
      CLR   DATAIN
      CALL  CLOCK_GETIR

      CLR   DATAIN
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      CALL  CLOCK_GETIR
      SETB  DATAEN

      RET
;*********************************************
CLOCK_GETIR:
      SETB  CLOCK             ;bir saat darbesi
      CLR   CLOCK
      RET
;*********************************************
BIT_BIT_GONDER:

      MOV   SAYAC,#07
TUR:

      RRC   A
      MOV   DATAIN,C
      CALL  CLOCK_GETIR
;*********************************************

INITIALIZE:

      MOV   BIR_BUF,#00H
      MOV   ON_BUF,#00H
      MOV   YUZ_BUF,#00H
      MOV   VOLT,#00H
      MOV   SAYAC,#00H
;----------------
```

```
      CLR   CLOCK             ;
      CLR   DATAIN            ;Kullanılan port bacaklarının
      CLR   DATAEN            ; ilk koşulları atandı
;----------------

      MOV   ADCCON1,#10000000B ;ADC ayarları atandı
      MOV   ADCCON2,#00B      ;

      MOV   ADCDATAH,#0H      ;ADC veri registerları s
      MOV   ADCDATAL,#0H      ;sıfırlandı
                              ;kalibrasyon
      CLR   ADCI
      MOV   ADCCON3,#00000001B
      JNB   ADCI,$
      CLR   ADCI
      MOV   ADCCON3,#00000011B
      JNB   ADCI,$
      CLR   ADCI

      MOV   SAYAC,#00         ;bazı sayaçlar
      MOV   ADC_SAYAC,#00

      MOV   VOLT,#00111111B   ;int-dsply en soluna yazılacak
                              ; sabit sıfır sayısı
;----------------

      ORL   PLLCON,#00000011B ;CORE CLOCK_GETIR PLL YARDIMIYLA
                              ;2.097152 MHz'E AYARLANDI.
      MOV   TMOD,#00000001B   ;TİMER-0 MOD-1 SEÇİLDİ.

      MOV   TH0,#0F9H         ;10mili saniye için gerekli olan değer
      MOV   TL0,#02CH         ;TH0 ve TL0'a atanır
      SETB  EA                ;TÜM INTERRRUPTLAR AÇILDI.
      SETB  ET0               ;TIMER-0 İNTERRUPT'I AÇILDI.

      SETB  TR0               ;TIMER-0 ÇALIŞTIRILIR.
;----------------
      RET
;*************************************************
SEV_SEG_TAB:                  ;Hanede görünecek olanı secen
                              ; tablodur

        DB  00111111B         ;0 Görünür
        DB  00000110B         ;1 Görünür
        DB  01011011B         ;2 Görünür
        DB  01001111B         ;3 Görünür
        DB  01100110B         ;4 Görünür
        DB  01101101B         ;5 Görünür
        DB  01111101B         ;6 Görünür
        DB  00000111B         ;7 Görünür
        DB  01111111B         ;8 Görünür
        DB  01101111B         ;9 Görünür
;*************************************************
END
```

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Ek Bilgi:Görüldüğü gibi Timer0_ISR ve Intelligent Display Sürme programlarında birkaç değişiklik ve kısa bir ADC okuma rutini eklendiğinde her türlü sensör,voltmetre v.b. uygulamalarda sorunsuz kullanabileceğimiz 8-bit çözünürlüklü bir ADC donanımı elde ettik.Bu uygulamanın ekranda görüntülenen sonucunu bir voltmetre ile test etmek isterseniz %2 gibi bir hata oranı ile karşılaşırsınız. Ancak sanıldığı gibi bu hata payı sizin donanım ve yazılımınıza değil voltmetrenize ait olacaktır.

9.3.12 ADuC814 Minikit ve 74HC595 ENTEGRESİNİN ORTAK KULLANIMI UYGULAMASI

Önbilgi: ADuC814 MiniKit'i bu uygulamadan önceki uygulamaların hepsinde kullandıktan sonra akım sürebilen "Port Pini" azlığı kolayca fark edilmektedir. Ancak ürünün en önemli özelliği adındanda anlaşılacağı gibi ADuC814'ün bir "Mini MicroConverter" olmasıdır.Sonuç olarak endüstri standardı 8051 yapısında hazırlanmış bir Mikrodenetleyici olan ADuC814 'ün asıl amacı ADC-DAC dönüşümünü yüksek rezolüsyonda yapabilmektir.Bu kullanım engeli dışındaki çok kapsamlı ve kullanıcı dostu özellikleri sayesinde ADuC814 kullanıcıyı başka bir denetleyici seçmek yerine bu engeli kaldırmaya yöneltmektedir.

Pin problemi en basit olarak 74HC595 entegresi ile çözümlenebilir.Bu entegre 8 bitlik bir shift register mantığıyla çalışmaktadır.En basit anlatım şekliyle denetleyicinizin sadece 3 pinini kullanarak bütün bir Port kazanabilirsiniz.

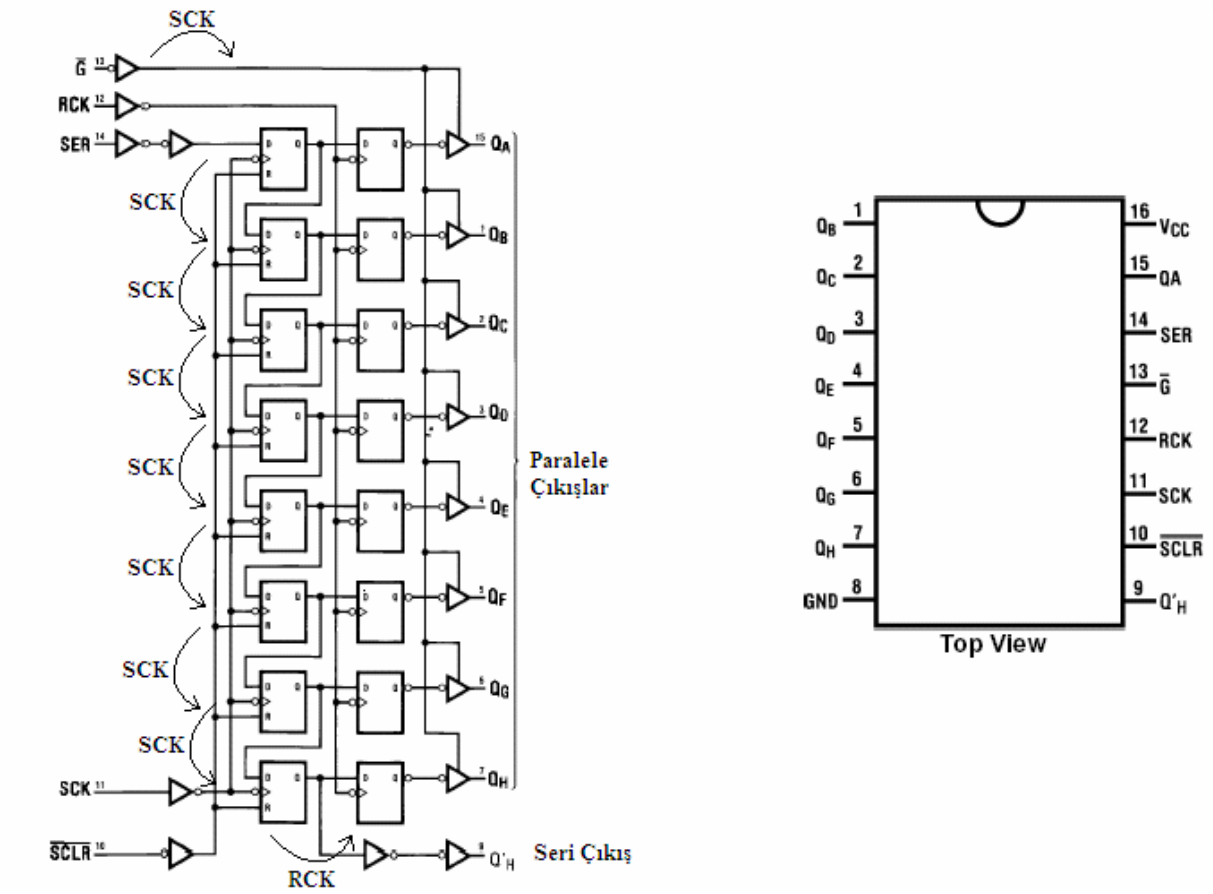
Bu entegrenin çalışma mantığı genel anlamda çok basittir. Entegrenin sürülmesinde kullanılan 3 port pini sırasıyla SCLK,RCLK ve SER pinlerini kontrol edecektir bu pinler:

SCLK: SHIFT CLOCK pinini tipik intelligent display sürerken kullandığımız clock mantığında olduğu gibi kullanacağız.Programın initialize bölümünde "0" olarak atadığımız bu pin ile her bir clock sinyali gönderdiğimizde SER pinine uygulamış olduğumuz logic değer 74HC595 entegresinin 0 numaralı register bitine yerleştirilir bu işlemden sonraki her bir Shift Clock darbesi durumunda SER bilgisi kendinden bir önceki register içeriğini bir sıra kaydırarak ilk register bitinine yerleşir.Ancak bu sırada entegrenin Q_A-Q_H pinlerindeki sayısal değerler değişmez.

SER: Bu pin yukarıda da anlatıldığı gibi her SCLK darbesi üretildiğinde içeriğini ilk register bitine aktarır.Örneğin bir bytelık port durumunu 74HC595 pinlerine atarken kullandığımız 8 SCLK darbesinin her birinden önce gereken pin değeri bu pine atanır.

RCLK: REGISTER CLOCK pini register bitlerinin doldurulması işlemi tamamlandıktan sonra bu bitleri 74HC595 çıkışlarına aktarmak için gereken clock darbesini yaratmakta kullanılır.

Sonuç olarak aşağıdaki şekilde görüldüğü gibi bit bilgileri önce registerlar arasında tek tek kaydırılır sonra da hepsi birden tek bir darbe ile pinlere atanır ancak bu darbe gelene kadar 74HC595 bacaklarındaki sayısal değer hiç değişmez.



Entegrenin giriş,çıkış ve clock pinleri dışındaki bacaklar ise

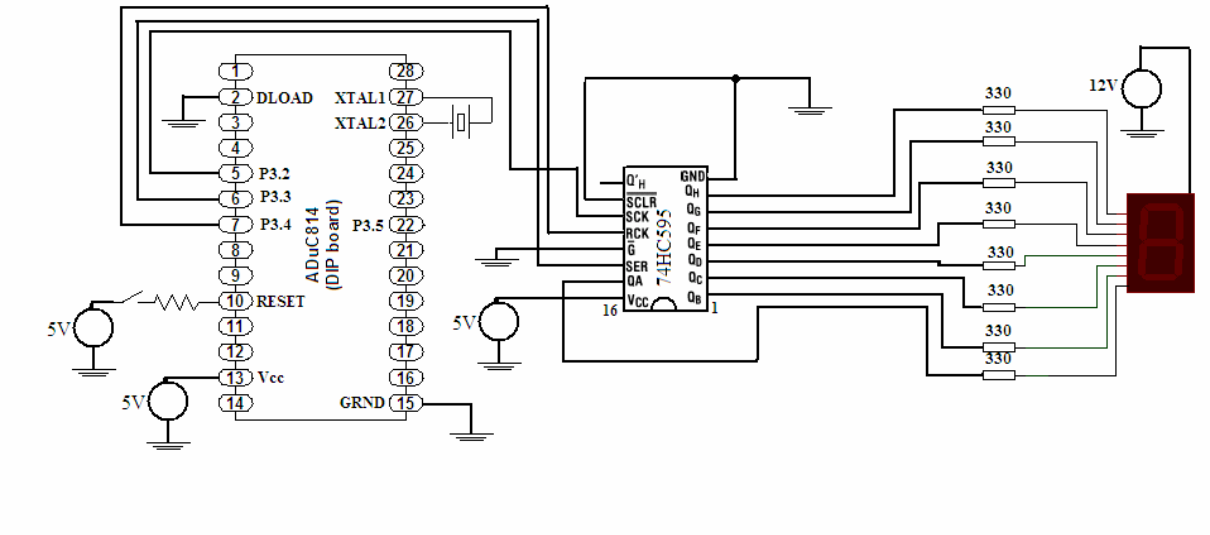
SCLR': Bu pin entegrenin çıkış portuna sürülmüş olan değeri resetlememize olanak sağlar aktif low çalıştığından lojik "1" iken etkisiz ardından "0" darbesi ile ise portların resetlenmesi işlemini gerçekleştirir eğer port un program akışı sırasında resetlenebilmesini istersek bu pin de ayrı bir port pinine bağlanabilir ancak bu entegrenin kullanım amacı bu uygulamada port pini sayısını arttırmak olduğundan mantıksız olacaktır.Bu uygulamada bu bacak 5V değere çekilmiştir

G': Tıpkı SCLR' gibi aktif low çalışan GATE pini bu entegeriyi tümden aktive eden pindir Bir birlerinden bağımsız port çoklayıcı entegreler kullanıldığında bu pinler aynı işlevli farklı entegrelerin seçilmesinde kolaylık sağlar ancak bu uygulamada entegre sürekli aktif tutulmak istendiğinden bu pin toprağa çekilmiştir.

Q'_H: Bu pin 74HC595 entegrelerinin birbirlerine kaskat biçimde bağlanarak daha büyük kapasiteli (16bit,24bit,32bit…) shift registerlar elde etmek için kullanılır

Bu uygulamada 74HC595 ile MiniKit kullanımını kolay kavrayabilmek açısından önceki uygulamalarda sözünü ettiğimiz 74HC595 ile 7-Segment Display Sürülmesi Uygulamasını gerçekleştireceğiz.

Devre Bağlantıları:



Program Kodu:

Bu program genel olarak 7447 ile 7-Segment Süren Programa birebir benzemektedir sadece haneyi yazma işlemi ve look-up table ve definition bölümü değişeceğinden bu bölümde değişen program parçaları gösterilmiştir. Programın diğer kısımları hakkında bilgi için 7447 ile 7-Segment Sürülmesi bölümüne başvurabilirsiniz.

Definition Bölümü;

```
SCK    EQU    P3.2        ;Programın anlaşılabilirliği açısından pinler
SER    EQU    P3.3        ;isimlendirildi
RCK    EQU    P3.4
```

Haneye Yazma İşlemi;

```
;*****************************************************
HANEYE_YAZ:

       MOV    A,SAN_BUF
       MOV    DPTR,#SEV_SEG_TAB       ;look up table'ın başlangıç adresi
                                      ; DPTR'ye atanır
       MOVC   A,@A+DPTR               ;A ya tablonun istenen değeri
                                      ;atanır
       CALL   BIT_BIT_GONDER

       SETB   RCK                     ;Bitlerin gönderilimi tamamlandığında
       CLR    RCK                     ;Bit bilgisinin port pinlerine atanması
                                      ;bir RCK darbesi ile sağlanır

       RET
;--------------------------------------

CLOCK_GETIR:
       SETB   SCK                     ;Her bit gönderilirken bir SCK darbesi
       CLR    SCK                     ; uygulanır
```

```
      RET

;-------------------------------------
BIT_BIT_GONDER:
      MOV   SAYAC,#08
TUR:
      RRC   A                       ;Bu işlem Porta atanacak olan değerin
      MOV   SER,C                   ;bit bit 74HC595 e gönderilmesidir
      CALL  CLOCK_GETIR

      DJNZ  SAYAC,TUR
      RET

;*******************************************
```

Look –Up Table;

```
;*****************************************
SEV_SEG_TAB:                          ;Hanede görünecek olanı secen tablodur
                                      ;Ortak Anot 7-SEG için düzenlenmiştir


        DB  11000000B                    ;0 Görünür
        DB  11111001B                    ;1 Görünür
        DB  10100100B                    ;2 Görünür
        DB  10110000B                    ;3 Görünür
        DB  10011001B                    ;4 Görünür
        DB  10010010B                    ;5 Görünür
        DB  10000010B                    ;6 Görünür
        DB  11111000B                    ;7 Görünür
        DB  10000000B                    ;8 Görünür
        DB  10010000B                     ;9 Görünür
;************************************************
```

Ek Bilgi: Bu uygulamada kullandığımız entegre QH' pini bir diğer 74HC595 in SER pinine bağlanıp bu entegrelerin SCK ve RCK pinleri ortak yapıldığında 16 bitlik bir shift register entegresi görebilir ve yine sadece 3 pinden kontrol edilmeye devam eder.Ancak kaskat bağlı entegre sayısı arttıkça donanım ve program akış hızı düşecektir çünkü her bit tek tek atanmaktadır.Bu yüzden ADuC serisinin yüksek port pini adetli üyelerini kullanmak daha mantıklı olacaktır.

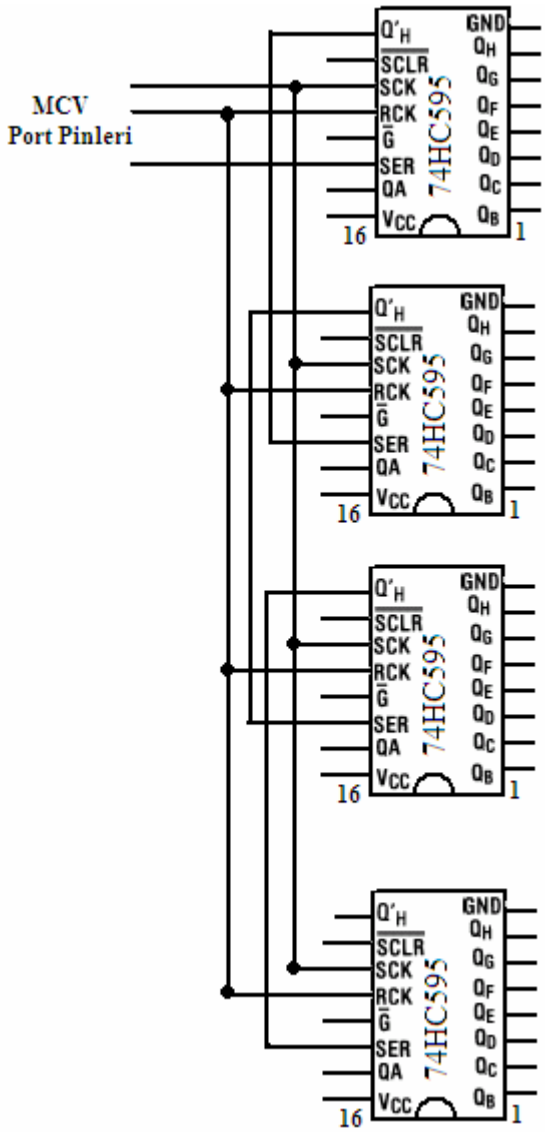
9.3.13 ÇOK HANELİ 7-SEGMENT UYGULAMASI

Önbilgi: Bir önceki uygulamada anlatılan tek hane sistemi 74HC595 entegreleri kaskat şekilde bağlandığında yine sadece 3 pin kullanılarak çok haneli bir modele rahatlıkla dönüştürülebilir.Bu uygulama için değişen olgular port buffer sayısının artması,devre eklemeleri ve görüntülenecek sayıların ihtiyacı olan bufferların özelliklerinin belirlenmesidir.

Bu uygulamada 4 adet 74HC595 entegresi ve 4 adet 7-Segment kullanarak bir saniye dakika sayacı yapacağız. Önceki uygulamalardan bildiğiniz gibi LedTech Intelligent Display'leri ile bu iş çok basit ve bizi karmaşık devre bağlantılarından kurtaracak nitelikteidi ancak bu uygulamadaki asıl amaç bir sayaç oluşturabilmek değil birden fazla port kontrolünü gerçekleştirmektir.

Burada öğreneceğiniz yöntem aynı anda bir çok çevresel sistemi kontrol ederken çekeceğiniz pin sıkıntısından nasıl korunabileceğinizi gösterecektir.

Devre Bağlantıları:



74HC595 entegrelerinin kaskat bağlantı şeklinde tüm SCK ve RCK pinleri ortak kullanılır.Ayrıca her Q'_H bacağı bir sonraki entegrenin SER bacağına bağlanarak entegrenin Q_H registerındaki bilgi bir sonraki SCK sinyalinde bir sonraki entegrenin Q_A registerına atanmak üzere ayarlanmış olur.

Elbetteki her entegrenin bu üç pin dışındaki diğer bağlantıları bir önceki uygulamada kullanılan standart yapıya göre yapılmalıdır.

Program Kodu:Önceki uygulamamızda yaptığımız gibi bu program kodunu da başka bir uygulamanın program kodu üzerinde ufak bir değişiklik yaparak hazırlayacağız.Bu tür bir çalışma mantığı bize bir program kütüphanesi hazırlamanın ne kadar yararlı bir yöntem olduğunu bir kez daha kanıtlamaktadır.

Şimdi Intelligent Display ile Dakika saniye sayıcı uygulamasındaki "DISPLAY_SUR" sub rutinini yeni devre ve port yapımıza göre güncelliyoruz tabi öncelikle intelligent display sürerken kullandığımız 3 port pinini definitions kısmında 74HC595 entegresinde kullandığımız şekilde isimlendiriyoruz.Portlara sürmemiz gereken sayılar bu programın önceki bölümlerinde SAN_BIR_BUF,SAN_ON_BUF,DAK_BIR_BUF ve DAK_ON_BUF isimli değişkenlerde

bulunmaktadır.En sona gelecek olan buffer ilk olarak porta yazdırılacağından en sağda görüntülenecek olan SAN_BIR_BUF değerinden göndermeye başlıyoruz

Definition Bölümü;

```
SCK   EQU   P3.2        ;Programın anlaşılabilirliği açısından pinler
SER   EQU   P3.3        ;isimlendirildi
RCK   EQU   P3.4
```

Display_Sur Altrutini;

```
;****************************************
DISPLAY_SUR:

      MOV   A,SAN_BIR_BUF
      MOV   DPTR,#SEV_SEG_TAB       ;look up table'ın başlangıç adresi
                                    ; DPTR'ye atanır
      MOVC  A,@A+DPTR               ;A ya tablonun istenen değeri
                                    ;atanır
      CALL  BIT_BIT_GONDER


      MOV   A,SAN_ON_BUF
      MOV   DPTR,#SEV_SEG_TAB       ;look up table'ın başlangıç adresi
                                    ; DPTR'ye atanır
      MOVC  A,@A+DPTR               ;A ya tablonun istenen değeri
                                    ;atanır
      CALL  BIT_BIT_GONDER


      MOV   A,DAK_BIR_BUF
      MOV   DPTR,#SEV_SEG_TAB       ;look up table'ın başlangıç adresi
                                    ; DPTR'ye atanır
      MOVC  A,@A+DPTR               ;A ya tablonun istenen değeri
                                    ;atanır
      CALL  BIT_BIT_GONDER


      MOV   A,DAK_ON_BUF
      MOV   DPTR,#SEV_SEG_TAB       ;look up table'ın başlangıç adresi
                                    ; DPTR'ye atanır
      MOVC  A,@A+DPTR               ;A ya tablonun istenen değeri
                                    ;atanır
      CALL  BIT_BIT_GONDER


                                    ;Tüm 32 bitin
      SETB  RCK                     ;gönderilimi tamamlandığında
      CLR   RCK                     ;Bit bilgisinin port pinlerine atanması
                                    ;bir RCK darbesi ile sağlanır

      RET
;---------------------------------------
```

```
CLOCK_GETIR:
      SETB  SCK                    ;Her bit gönderilirken bir SCK darbesi
      CLR   SCK                    ; uygulanır
      RET

;--------------------------------------
BIT_BIT_GONDER:
      MOV   SAYAC,#08
TUR:
      RRC   A                      ;Bu işlem Porta atanacak olan değerin
      MOV   SER,C                  ;bit bit 74HC595 e gönderilmesidir
      CALL  CLOCK_GETIR

      DJNZ  SAYAC,TUR
      RET

;*********************************************
```

Look –Up Table;

```
;*****************************************
SEV_SEG_TAB:                        ;Hanede görünecek olanı secen tablodur
                                    ;Ortak Anot 7-SEG için düzenlenmiştir


        DB  11000000B                   ;0 Görünür
        DB  11111001B                   ;1 Görünür
        DB  10100100B                   ;2 Görünür
        DB  10110000B                   ;3 Görünür
        DB  10011001B                   ;4 Görünür
        DB  10010010B                   ;5 Görünür
        DB  10000010B                   ;6 Görünür
        DB  11111000B                   ;7 Görünür
        DB  10000000B                   ;8 Görünür
        DB  10010000B                   ;9 Görünür
;************************************************
```

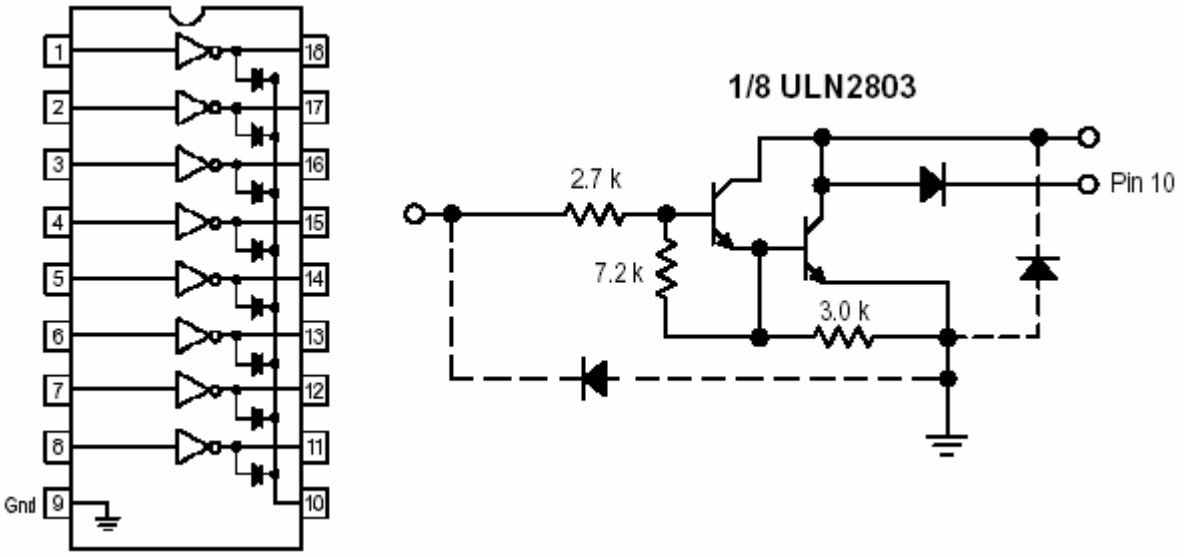
Ek Bilgi: Kaskat şekilde bağlanmış 74HC595'ler kullanımda büyük kolaylık sağlamalarına karşın entegre sayısı arttıkça program işleyişinde yavaşlamalara neden olmaktadır bu sebeple yüksek port adetli MCV kullanılması daha verimli görünmektedir ayrıca ileriki uygulamalarda kullanacağımız 74HC574 ve 74HC138 entegreleri ile Memory Mapped I/O yöntemini kullanarak bir MCV ile bir çok çevresel ürünü kontrol etmek mümkün olacaktır.

9.3.14 BÜYÜK KARAKTER BOYUTLU 7-SEGMENT SÜRÜLMESİ

Ön Bilgi: Standart boyutlu bir 7-Segment Display için her bir segment içerisinde bir adet LED bulunduğunu söylemiştik.Ancak daha büyük karakter boyutlu ürünlerde segment başına düşen LED sayısı ve bu LEDlerin dizilimi değişmektedir.Sonuç olarak bu yeni diplay ürünlerine gereken gerilim ve akım miktarları hayli yüksek olmaktadır.Örneğin LedTech ürünlerinden olan 3 inç boyutlu tek hane 7-Segment Display her segmentinde 6 adet LEDi birbirlerine seri bağlanmış bir biçimde içermektedir.

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

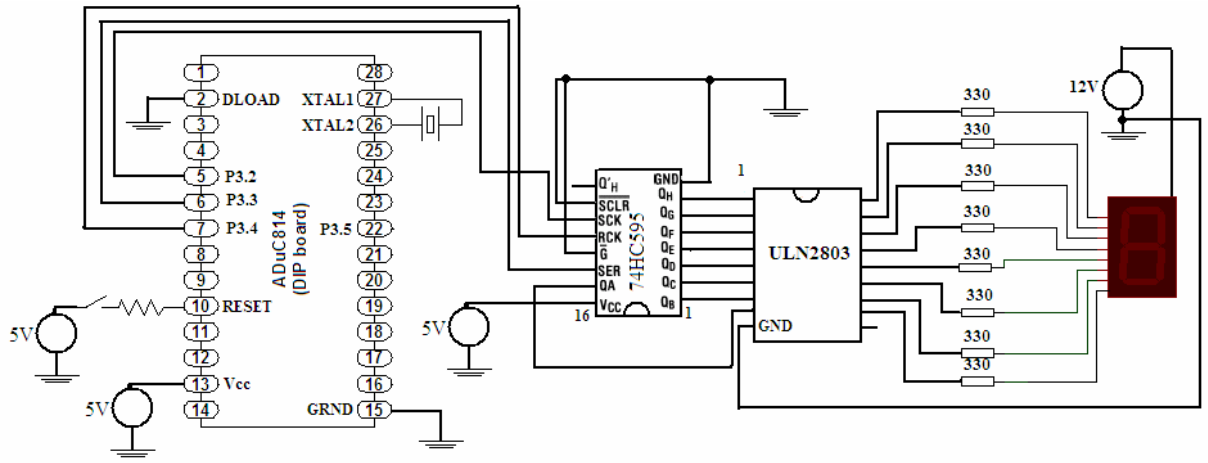
Bu displayler sürülürken programlama açısından standart 7-Segment sürme işlemine birebir örtüşmesine rağmen devresi kurulurken bir adet "Darlington Transistör" dizi entegresi kullanmak gerekmektedir.ULN2803 entegresi 8 pin desteklemesi açısından tam bu tipte bir uygulamaya göre hazırlanmış bir Darlington Transistör Dizisi'ni içermektedir.



Yukardaki şekilde görüldüğü üzere entegrede temel olarak her giriş bir "NOT" kapısı ve iki adet uçuca eklenmiş transistorden oluşmaktadır.Entegrede bir besleme bacağı olmamasından anlaşılmalıdır ki kullanacağımız 7-Segment ortak anot özellikli olmalıdır.Böylece transistör ile anahtarlama yapıldığında 7-Segmentin beslemesinden kaynağını alan ve segmentlerinden geçen akım bu transistörlerden geçerek toprağa akacaktır.Transistörün özelliğine göre baz bacağına "1" uygulandığında akım geçirecektir ancak "NOT" kapısı nedeniyle bir segmentin ışık vermesi için port pinlerimizden lojik "0" uygulamak gerekmektedir.

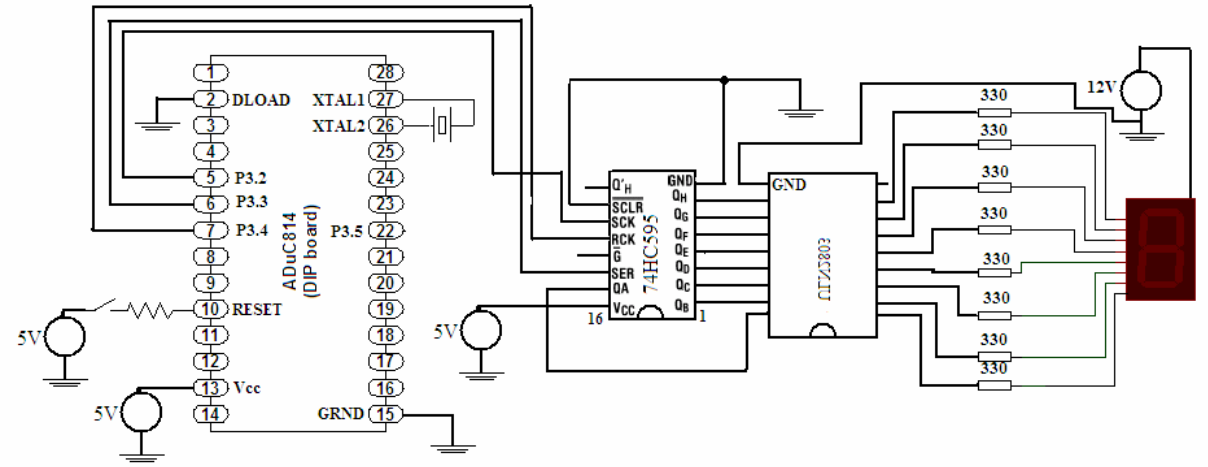
Bu teknik bilgilerin bizim uygulamamızda kullanacağımız kısmı özetlemek gerekirse 74HC595 entegresinin 7-Segment 'e bağlanacak olan pinleri önce ULN2803' de bulunan giriş pinine bağlanmalı sonra bu giriş pinine karşılık gelen çıkış pini istenilen segment pinine bağlanmalıdır.

Devre Bağlantıları:



Program Kodu: Bu uygulamanın program kodunun yazılımı açısından standart boydaki bir ortak anot 7-Segment sürülmesinden hiçbir farkı yoktur bu sebeple "Tek Hane 7-Segment Sürülmesi" isimli önceki uygulamanın program kodu bu uygulamada da çalışacaktır

Devre Bağlantıları:



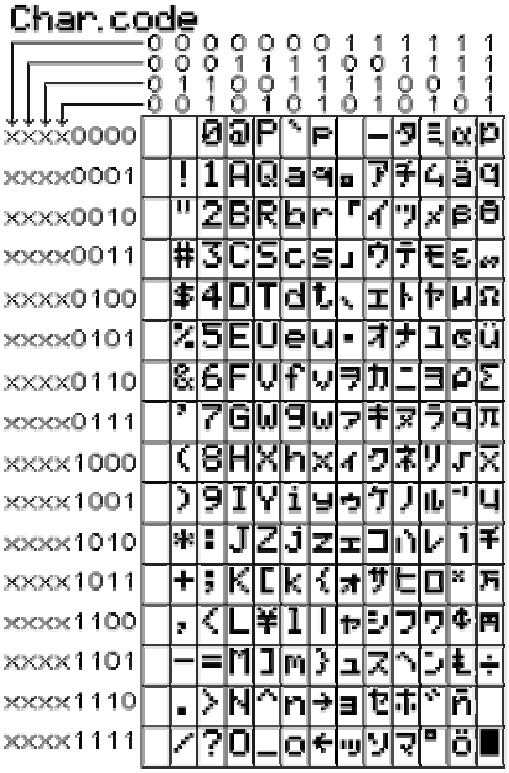
Ek Bilgi: ULN2803 entegresinin bu uygulamada kullanmadığımız 10 numaralı bacağı röle sürme işlemi gerçekleştirilirken rölenin bobin bacaklarından birinde kullanılır.Bir sonraki uygulamada işleyeceğimiz bu yöntem ile 8 rölenin aynı anda sürülmesi mümkün olacaktır.

9.3.15 LCD UYGULAMASI

LCD göstergeler ortaya çıktıkları 1970'li yılların başından beri popülaritelerini hiç yitirmemiş ve neredeyse her tür cihazda boy göstererek kendini kanıtlamışlardır. Bu popülarite iki önemli nedenden dolayı hiç azalmamıştır, bunlardan ilki düşük akım gereksinimi, diğeri ise kaplanan hacmin kalınlık bazında çok az olmasıdır. LED göstergelerin pabucunu dama bu özellikler atmaktadır. Fakat LCD'lerin önemli dezavantajları LED'lere göre daha zor sürme prensipleri ve parlak ışık altında silik görünmeleridir.[1]

LCD göstergeleri sürmek için BACKPLANE denilen ortak arka yüze ve her bir segmente birbiriyle 180 derece faz farkı olan 150 – 500 Hz değerinde kare dalga bir sinyal tatbik etmek gerekmektedir. Aydınlanması istenen segmente bu sinyal uygulanır. Bu sayede LCD sıvısı polarize olarak titreşir, buda göze görünür hale gelmesi demektir. Göze görünmemesi istenen segment ise sinyal uygulanmadığı için polarize olmaz ve görünmez. Tüm LCD ler kare dalga sinyale ve özel sürücü entegrelere ihtiyaç duyarlar.

LCM'ler akıllı göstergeler olarak anılırlar. İnce bir baskılı devre üzerine SMD(Surface Mount Device) tekniği ile monte edilmiş bir sürücü entegre bloğu, alfanümerik LCD gösterge, ve yapıyı sağlamlaştırmak için metal bir çerçeve LCM Modülleri oluştururlar. LCM'ler genellikle mikroişlemciler için tasarlanmış bir giriş/çıkış bus'ı içerirler. Bu bus yardımıyla LCM üzerinde her türlü bilgiyi göstermek mümkündür. LCM göstergeler 1x8, 1x16, 2x16, 4x20, 1x40, 2x40 satır/karakter olarak üretilmektedir. Bu modüllerin çoğunda üreticiler Hitachi firmasının HD44780U entegresini LCM sürücü entegresi olarak kullanmaktadırlar, bu da bir noktaya kadar bu tip mödüllere bir standart getirmektedir.



Şekil 1 **ASCII Tablosu**

[1] LCD ve LCM ile ilgili bilgiler: Türk Amatör Telsiz Gazetesi http://www.antrak.org.tr - Barbaros Aşuroğlu

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Sürme devresinin bir faydası da veri transferini ASCII standardını kullanarak yapmasıdır. (Şekil 1) ASCII standardında her karakterin bir sayısal karşılığı vardır. Mesela ekrana 'A' karakterini yazdırmak için ikilik düzende "01000001" yazdırmak gerekmektedir. Bilgisayarlar ve birçok elektronik cihazlar bu standardı kullandığı için sürme devresine bir yazı yazdırmak çok daha kolaydır. Programcının yazdırmak istediği karakteri derleyiciye vermesi yeterli olacaktır. Bir tabloya gerek kalmadan derleyici istenilen karakterin ASCII karşılığını ilgili yere koyacaktır.

Tablo 1. **GWMSM160208 için bacak bağlantıları.**

PIN NO	Sembol	Görev
1	VSS	Toprak
2	VDD	+5V Besleme (3V lada çalışabilir)
3	V0	Parlaklık ayarı
4	RS	Register seçim sinyali
5	R/W	Okuma/ Yazma seçim sinyali
6	E	Aktif sinyali
7	DB0	Data 0
8	DB1	Data 1
9	DB2	Data 2
10	DB3	Data 3
11	DB4	Data 4
12	DB5	Data 5
13	DB6	Data 6
14	DB7	Data 7
15	A	+4.2V İç aydınlatma (LED)
16	K	LED için toprak

Sürme devresinin genelde 16 tane girişi vardır. Bunlardan 8 tanesi veri iletişimi için diğerleri ise beslememe ve kontrol sinyalleri içindir. Bağlantı şekli standart olmamakla birlikte kullanılan LCD panel için Tablo 1'de verilmiştir. 15 ve 16 numaraları bağlantılar ekranın dâhili aydınlatması içindir. Bazı LCD'lerin üzerinde aydınlatma olmayabilir veya aydınlatma girişleri LCD'nin başka bir yerinden eklenmiş olabilir.

RS (Register Select):
RS ile LCD ye yollanan bilginin bir komut mu verimi olduğu kontrol edilir.
"0" (düsük) LCD panelin komut,
"1" (yüksek) LCD panelin veri için seçilmiş olur.

R/W (Read/Write):
R/W ile LCD di yazma veya okuma moduna getirilir.
"0" (düsük) LCD panelin yazma modunda,
"1" (yüksek) LCD panelin okuma modunda seçilmiş olur.

E (Enable) :
LCD nin girişleri istendiği gibi hazırlandıktan sonra E ucuna bir darbe uygulayarak LCD nin işleme başlaması sağlanır.

VO (Parlaklık) :
Bu giriş sayesinde LCD'nin parlaklık ayarı yapılır. Bir pot sayesinde (10K) parlaklık ayarı değişken yapılabileceği gibi sabit direnç ile istenilen parlaklık değerine sabitlenebilir.

DATA (DB0-DB7) :
Veri girişinin yapılacağı bacaklardır. 8.bitlik transferde verinin transferi için hepsi kullanılırken, 4. bitlik transferde üst ağırlıklı 4 bit kullanılır. Düşük ağırlıklı 4 bit ise topraklanılarak çevreden gelecek gürültülerden korunmalıdır.

LCD ye güç verdikten sonra sürme devresi amacımıza yönelik olarak ayarlanmalıdır. Ayarların yapılabilmesi için sürme devresinde tanımlanmış özel komutlar kullanılır. Bu komutların yanında, sürme devresi ekran temizleme, istenilen adrese gitme, istenilen adresteki bilgiyi okuma, ekranı kaydırma gibi işlemleri de yapabilir.

Bu ayarlardan biri, giriş bilgisinin 4 bit mi 8 bit mi olacağıdır. Eğer bacak sayısında sorun varsa 8 bit yerine 4 bitlik moda çalıştırılarak 4 bit ten tasarruf edilebilir. Bu şekilde çalışmada yollanacak 8 bit sırası ile 4 bitlik iki parça şeklinde sürme devresine verilmelidir. Sürme devresi bu iki dört biti birleştirerek 8 bitlik moda çevirir.

LCD'nin bu işlemleri yapabilmesi için belirli bir süreye ihtiyacı vardır. Bu süre dolmadan yollanacak yeni komutlar işlenemeyecektir. Mikrokontrolcü içinde bekleme alt programları yazılarak bu sorunun çözülebilir. Fakat bu çözümde, LCD'nin bozulması veya mikrokontrolcünün kristalinin değişmesi durumunda programımız doğru bir şekilde çalışamaz. İkinci çözüm ise sürme devresinin özelliği olan meşgul bayrağını kontrol etmektir. Sürme devresi verilen komut bittikten sonra eğer isteniyorsa meşgul bayrağını indirerek bir sonraki komuta hazır olduğunu belirtir. RS=0 ve R/W=1 yapılırsa veri hattının son biti (7.Bit) meşguliyet durumunu verecektir. Eğer mikrokontrolcünün çıkışları yeterliyse bu özellik kullanılmalıdır: Bu sayede hem programın dayanıklılığı artırılmış olur hem de bekleme alt programlarına ihtiyaç olmadığı için ana program kısalacaktır. Zaman bekleyerek yapılan kontrolde tek fayda R/W ucu direk olarak topraklanarak bir giriş/çıkış'tan kar edilebilmesidir.

LCD KOMUTLARI:

	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1 Ekranı temizler	0	0	0	0	0	0	0	0	0	1
Çalışma zamanı : 64ms										
Açıklama : Tüm ekranı temizler ve ilk adresse döner. (adr0)										
2 En başa döner	0	0	0	0	0	0	0	0	1	*
Çalışma zamanı : 64ms										
İlk adresse geri döner (adr0).										
3 Giriş modu ayarı	0	0	0	0	0	0	0	1	I/D	S
Çalışma zamanı : 40us										
Veri yazıp okurken cursorun hareket yönünü ve otomatik olarak kayıp kaymayacağını ayarlar.										
4 EKRAN AYARI	0	0	0	0	0	0	1	D	C	B
Çalışma zamanı : 40us										

Ekranı aç-kapa(D), Cursor aç-kapa(C) ve cursorun yanıp sönmesi(B)

5 Kaydırma 0 0 0 0 0 1 S/C R/L * *
Çalışma zamanı : 40us
Cursoru ve ekranı hafıza değerlerini değiştirmeden sağa veya sola kaydırma.

6 Genel Ayarlar 0 0 0 0 1 DL N F * *
Çalışma zamanı : 40us
Data uzunluğu(DL), satır sayısı(N) ve font büyüklüğü(F)

7 CG RAM Ayarla 0 0 0 1 (CG RAM address)
Çalışma zamanı : 40us

8 DD RAM Ayarla 0 0 1 (DD RAM address)
Çalışma zamanı : 40us

9 Meşgul bayrağı 0 1 BF (AC)
Çalışma zamanı : 0us
Meşgul bayrağını okumak için kullanılır. (BF) CGRAM veya DDRAM adreslerinin değeri(AC).

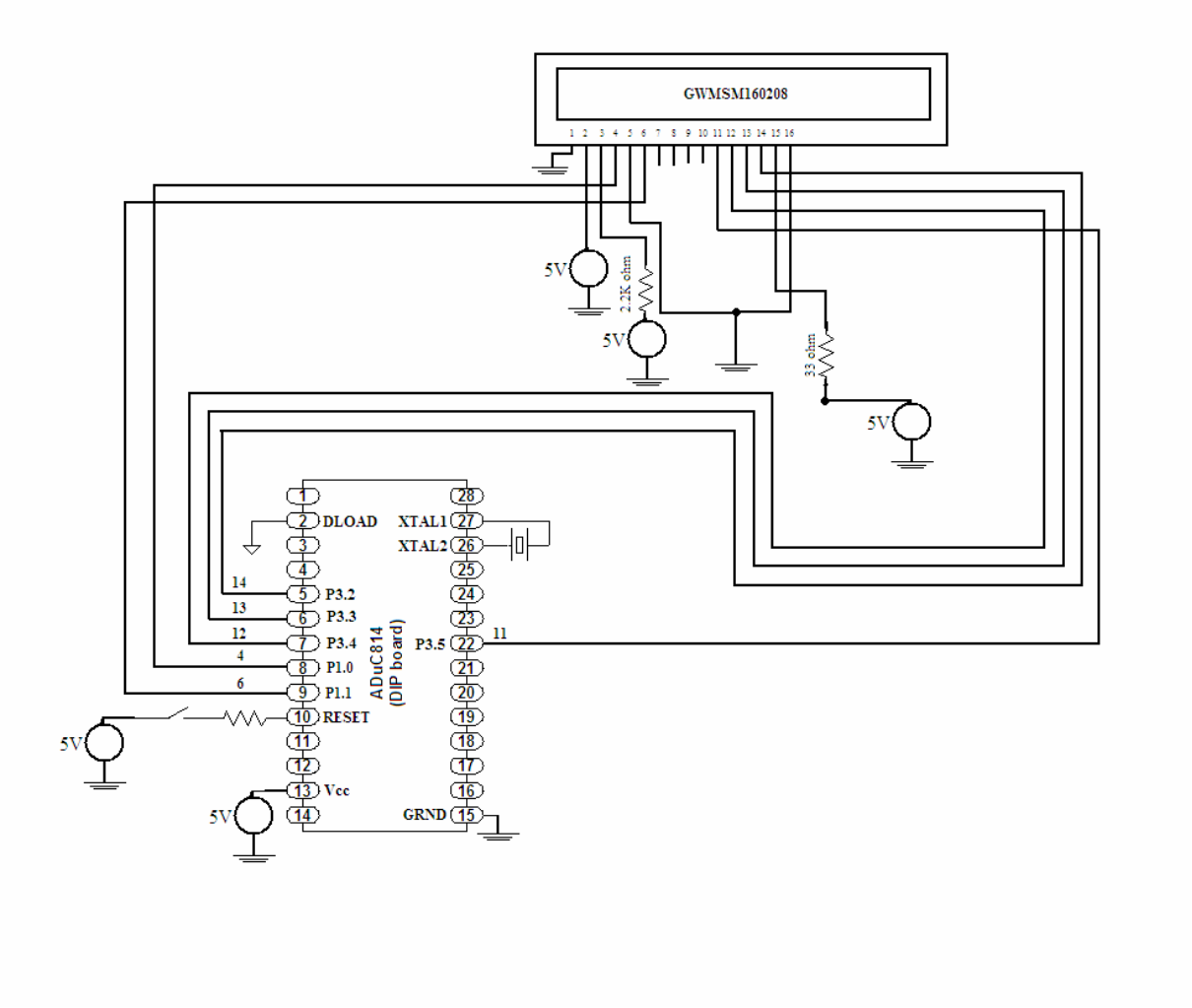
10 Veri yazma 1 0 (yazılacak veri)
Çalışma zamanı : 40us

11 Veri okuma 1 1 (okunan veri)
Çalışma zamanı : 40us

I/D = 1: Artırma (+1) I/D = 0: Azaltma(-1)
S = 1: Kendiliğinden kayar
S/C = 1: Ekran
S/C = 0: Cursor
R/L = 1: Sağa
R/L = 0: Sola
N = 1: 2 satır N = 0: 1
F = 1: 5x10 noktalı F= 0: 5X7 noktalı
BF = 1: İşlem Devam ediyor.
BF = 0: Yeni işlem yapılabilir.

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Devre Bağlantıları:



Program Kodu:

```
$MOD814

ON_MSN          DATA  06FH      ;on milisaniyeyi saymak için
                                ;gerekli sayaç tanımlandı.
YUZ_MSN         DATA  06EH      ;yüz milisaniyeyi saymak için
                                ;gerekli sayaç tanımlandı.
SANIYE          DATA  06DH      ;saniyeyi saymak için gerekli
                                ;sayaç tanımlandı.

BYTE_SAYACI     DATA  06BH

LCD_D7          EQU   P3.2
LCD_D6          EQU   P3.3
LCD_D5          EQU   P3.4
LCD_D4          EQU   P3.5
LCD1_EN         EQU   P1.1
LCD_RS          EQU   P1.0
```

```
      LCD_DD_RAM_ADRES_SET    EQU   10000000B

      LCD_5MSN_SAYACI         DATA  08H
      LCD_50USN_SAYACI  DATA  09H



      CSEG

      ORG   0000H                   ;MAIN_PROGRAM'ın başalngıç adresi

      JMP   MAIN                    ;MAIN_PROG'a atla.

      ORG   000BH                   ;Interrupt oluştuğu anda başalnıcak adres.

      JMP   TMR0_ISR                ;Interrupt geldiği zaman TMR_ISR'e atla.

      ORG   0060H                   ;programın hafızada yer aldığı ye ile
                                    ;interrupt'ın hafızdaki yeri ayırmak için.
MAIN:

      MOV   SP,#2FH

      CALL  INITIALIZE


TEKRAR:

      JMP   $

;*************************************************************



TMR0_ISR:

      MOV   TL0,#02CH         ;gerekli değerler yüklenir.
      MOV   TH0,#0F9H         ;Döngü içerisinde timer-0'a


      PUSH  ACC               ;
      PUSH  B                 ;
      PUSH  PSW               ;Program akışında gerekli olan
                              ;registerlar stack'e atılarak
      PUSH  DPH               ;Saklanır.
      PUSH  DPL               ;



      INC   ON_MSN            ;ON_MSN değişkeni bir artırılır.
      MOV   A,ON_MSN          ;Karşılaştırma yapmak için ON_MSN'e A'ya
                              ; aktarılır.
      CJNE  A,#0AH,GERIDON    ;Eğer A (ON_MSN) 10'a ulaştıysa devam et,yoksa
```

```
                              ; GERIDON'e atla.
      MOV   ON_MSN,#00H       ;Bir sonraki dögü için ON_MSN'yi sıfırla hazır
                              ; tut.


      INC   YUZ_MSN                ;YUZ_MSN değişkenini bir arttır.
      MOV   A,YUZ_MSN              ;Karşılaştırma yapmak için YUZ_MSN'e
                                   ; A'ya aktarılır.
      CJNE  A,#0AH,GERIDON         ;Eğer A (YUZ_MSN) 10'a ulaştıysa devam
                                   ; et,yoksa GERIDON'e atla.
      MOV   YUZ_MSN,#00H           ;Bir sonraki dögü için YUZ_MSN'yi
                                   ; sıfırla hazır tut.


      CALL  LCD_MESAJ_BELIRLE      ;Her saniyede bir LCD mesajı yenilenir

      INC   SANIYE                 ;SANIYE değişkenini bir arttır.
      MOV   A,SANIYE               ;Karşılaştırma yapmak için SANIYE'e A'ya
                                   ;aktarılır.
      CJNE  A,#3CH,GERIDON         ;Eğer A (SANIYE) 60'a ulaştıysa devam
                                   ; et,yoksa GERIDON'e atla.
      MOV   SANIYE,#00H            ;Bir sonraki dögü için YUZ_MSN'yi
                                   ; sıfırla hazır tut.

GERIDON:

      POP   DPL               ;
      POP   DPH               ;Program akışında gereklli olan registerları
                              ;ilk değerleri stack'ten geri çağrılır,
      POP   PSW               ;programın akışına devam edilir.
      POP   B                 ;
      POP   ACC               ;

RETI                          ;MAIN_PROG'a geri dön.


;***********************************************


INITIALIZE:

      MOV   TMOD,#00000001B         ;TİMER-0 MOD-1 SEÇİLDİ.

      MOV   TH0,#0F9H         ;10mili saniye için gerekli olan değer
        MOV TL0,#02CH         ;TH0 ve TL0'a ataanır
      SETB  EA                ;TÜM İNTERRRUPTLAR AÇILDI.
      SETB  ET0               ;TİMER-0 İNTERRUPT'I AÇILDI.
      CALL  LCD_INIT
      SETB  TR0               ;TİMER-0 ÇALIŞTIRILIR.

      RET

;*************************************************
MSJ_SELIM_TABLO:                  DB '  SELIM DILMAC  '
MSJ_ELEKTRO:                      DB '    ELEKTRO     '

;*************************************
LCD_INIT:
```

```
            MOV   R1,#022h
 DLY1:
            MOV   R2,#0FFh
            DJNZ  R2,$              ; 100 msn gecikme
            DJNZ  R1,DLY1

      CLR LCD1_EN
      CLR LCD_RS

      CALL BEKLE_5MSN
      CALL BEKLE_5MSN
      CALL BEKLE_5MSN         ;15 msn beklendi

      MOV A,#3FH
      CALL KOMUT_GONDER
      CALL BEKLE_5MSN

      MOV A,#3FH
      CALL KOMUT_GONDER
      CALL BEKLE_50USN
      CALL BEKLE_50USN        ;100 usn beklendi

      MOV A,#32H
      CALL KOMUT_GONDER

      MOV A,#2FH
      CALL KOMUT_GONDER

      MOV A,#0CH
      CALL KOMUT_GONDER

      RET

;*************************************
BEKLE_50USN:
      MOV LCD_50USN_SAYACI,#4
      DJNZ LCD_50USN_SAYACI,$
      RET
;*************************************
BEKLE_5MSN:
      MOV LCD_5MSN_SAYACI,#10
B5M_NL:
      MOV LCD_50USN_SAYACI,#28
      DJNZ LCD_50USN_SAYACI,$
      DJNZ LCD_5MSN_SAYACI,B5M_NL
      RET
;*************************************
KOMUT_GONDER:
      CLR LCD_RS
      CALL BYTE_GONDER

      RET
;*************************************
DATA_GONDER:
      SETB LCD_RS
      CALL BYTE_GONDER

      RET
```

```
;**************************************
BYTE_GONDER:
      RLC A
      MOV LCD_D7,C
      RLC A
      MOV LCD_D6,C
      RLC A
      MOV LCD_D5,C
      RLC A
      MOV LCD_D4,C

      PUSH ACC
;---
      SETB LCD1_EN
      NOP
      NOP

      CLR  LCD1_EN
;---------
      POP ACC

      RLC A
      MOV LCD_D7,C
      RLC A
      MOV LCD_D6,C
      RLC A
      MOV LCD_D5,C
      RLC A
      MOV LCD_D4,C
;---
      SETB LCD1_EN
      NOP
      NOP
      CLR  LCD1_EN

;---------
      CALL BEKLE_50USN

      RET
;**************************************

LCD_1ST_SATIRA_GONDER:
      MOV A,#00
      CALL LCD_DD_RAM_ADRES

      MOV BYTE_SAYACI,#16
      MOV R0,#70H
LCD1SG_NB:
      MOV A,@R0
      CALL DATA_GONDER
      INC R0
      DJNZ BYTE_SAYACI,LCD1SG_NB
      RET
;************************************************

MESAJI_LCD_BUFLARA_YUKLE:
      MOV R0,#70H
      MOV BYTE_SAYACI,#16
```

```
MLBY_NB:
      CLR A
      MOVC A,@A+DPTR
      MOV @R0,A
      INC R0
      INC DPTR
      DJNZ BYTE_SAYACI,MLBY_NB

      RET
;************************************************

LCD_DD_RAM_ADRES:
      ADD A,#LCD_DD_RAM_ADRES_SET
      CALL KOMUT_GONDER
      RET

;******************************
LCD_MESAJ_BELIRLE:
      MOV   DPTR,#MSJ_ELEKTRO
      CALL  MESAJI_LCD_BUFLARA_YUKLE
      CALL  LCD_1ST_SATIRA_GONDER
      MOV   DPTR,#MSJ_SELIM_TABLO
      CALL  MESAJI_LCD_BUFLARA_YUKLE
      CALL  LCD_2ND_SATIRA_GONDER
      RET

;**********************
LCD_2ND_SATIRA_GONDER:
      MOV A,#40
      CALL LCD_DD_RAM_ADRES

      MOV BYTE_SAYACI,#16
      MOV R0,#70H
LCD2SG_NB:
      MOV A,@R0
      CALL DATA_GONDER
      INC R0
      DJNZ BYTE_SAYACI,LCD2SG_NB
      RET
;************************************************

END
```