## Bu uygulamada Stellaris® LM4F120 LaunchPad kartının pinlerine bağlanmış 4’lü 7 segment display kontrol edilmektedir. 4’lü 7 Segment display’in 8 adet segment, 4 adet ise satır seçme (GND) pinleri bulunmaktadır. 8 adet segment pinlerinin 7’si segmentler, 1’i de nokta segmenti içinidir. Ve bu pinlerin tamamı 4 adet farklı 7 segment displayde paralel bağlanmış şekildedir.

## Aksi takdirde tüm segmentlerin kontrolü için 4 \* 8 = 32 adet pin kullanılmak zorunda kalınacaktır. Bu sayede kartımızın pinlerinden tasarruf edilerek sadece görüntülemek istenilen segment seçildiğinde o segmentin LED’leri yanmaktadır.

## Fakat bu yöntemde her seferinde yanlızca 1 segment seçilebildiği için o an seçilen segment yanmakta ve diğerleri sönük durumda kalmaktadır. Bunu aşabilmek için çok hızlı bir şekilde segment seçimlerini güncellemek gerekmektedir. Eğer çok hızlı bir şekilde segmentler güncellenirse bu frekans insan gözünün algılayabileceğinden daha yüksek olduğu için tüm segmentleri yanıyomuş gibi görürüz. Bu yönteme satır tarama yöntemi denilmektedir.

|  |  |
| --- | --- |
| **Stellaris Port**  **pinleri** | **4-lü 7-Segment pinleri**  **(5461AS)** |
| J1[1] PB5  J1[2] PB0  J1[3] PB1  J1[6] PB4  J3[6] PE1  J3[7] PE2  J3[8] PE3  J4[2] PB3  J2[1] PB2  J2[2] PE0 (P30)  J2[6] PB6 | C19 - 11 (F)  B21 - 10 (A)  D59 - 7 (B)  G16 - 1 (E)  A39 - 8 (GND)  A38 - 9 (GND)  A35 - 12 (GND)  J18 - 2 (D)  I56 - 4 (C)  I40 - 6 (GND)  I58 - 5 (G) |

## rect3430.png

## Uygulama 1: TIMER 0 ve TIMER 1

startup\_gcc.c

|  |
| --- |
| **1) include kisimlarindan sonra asagidaki fonksiyon prototiplerini ekleyin**  **extern** **void** timer\_0\_handler();  **extern** **void** timer\_1\_handler();  **2) // The vector table yazan kisimdaki dizide aşağıdaki iki satırda:**  IntDefaultHandler, // Timer 0 subtimer A  IntDefaultHandler, // Timer 1 subtimer A  **IntDefaultHandler yerine ilgili fonksiyonuları yazin**  timer\_0\_handler, // Timer 0 subtimer A  timer\_1\_handler, // Timer 1 subtimer A |

main.c

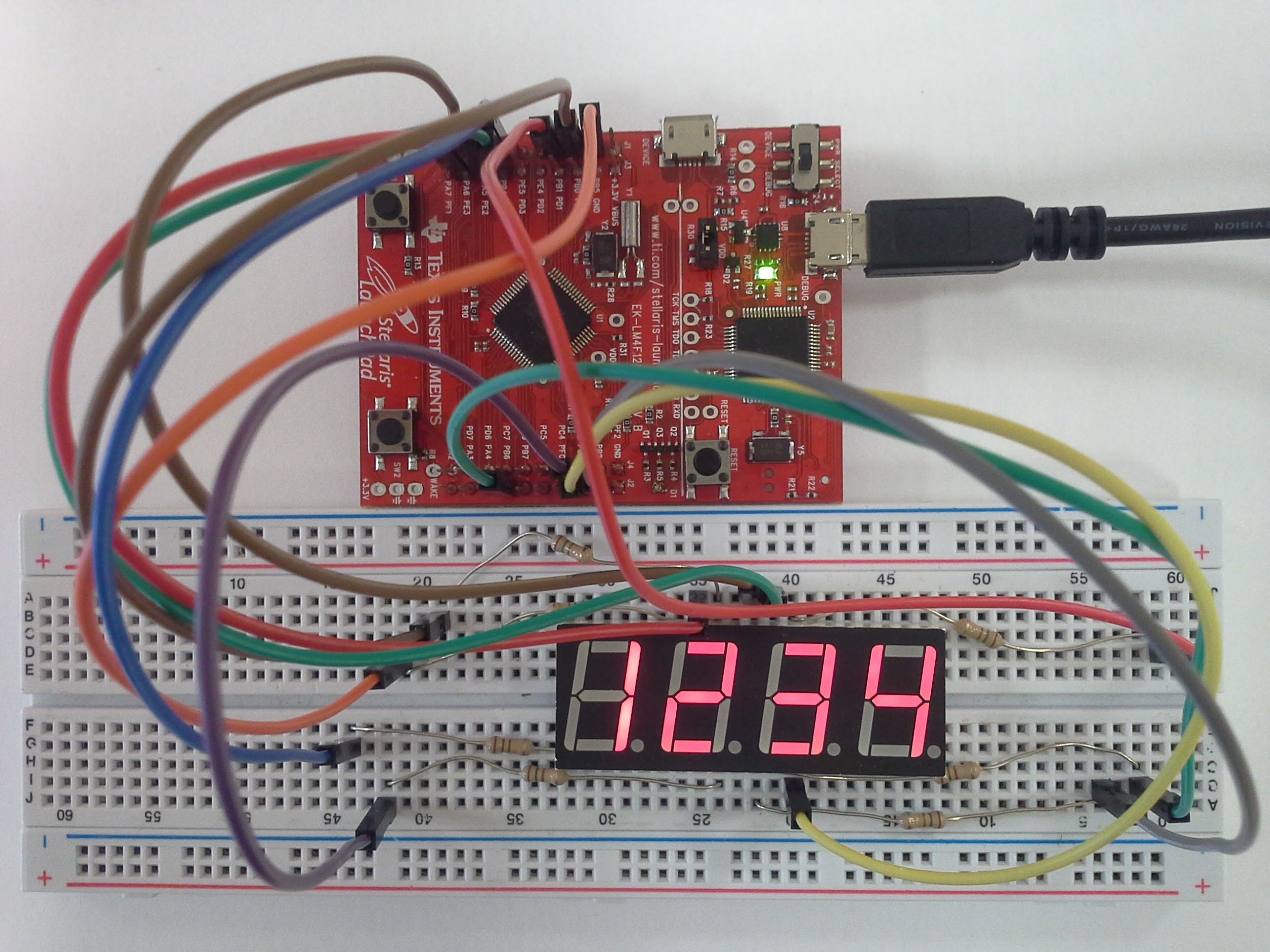
|  |
| --- |
| /\*\*  \* ornekte timer 0 ve timer 1 kesmesi kullaniliyor.  \*/   #include <stdint.h> #include "inc/lm4f120h5qr.h"  #include "inc/hw\_ints.h" #include "inc/hw\_memmap.h" #include "inc/hw\_types.h" #include "driverlib/interrupt.h" #include "driverlib/uart.h" #include "driverlib/rom.h" #include "driverlib/sysctl.h" #include "driverlib/timer.h" #include "utils/uartstdio.h"  **static** **inline** **void** disable\_interrupts() {  asm("CPSID I"); }  **static** **inline** **void** enable\_interrupts() {  asm("CPSIE I"); }  **static** **inline** **void** wait\_for\_interrupt() {  asm("WFI"); }  **void** init\_port\_B() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOB;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTB\_DIR\_R |= 0xFF;  GPIO\_PORTB\_AFSEL\_R &= ~0xFF;  GPIO\_PORTB\_DEN\_R |= 0xFF; }  **void** init\_port\_E() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOE;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTE\_DIR\_R |= 0b01110;  GPIO\_PORTE\_AFSEL\_R &= ~0b01110;  GPIO\_PORTE\_DEN\_R |= 0b01110; }  **void** init\_timer\_0() {  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);   TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);  TimerLoadSet(TIMER0\_BASE, TIMER\_A, SysCtlClockGet());   IntEnable(INT\_TIMER0A);  TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);   TimerEnable(TIMER0\_BASE, TIMER\_A); }  **void** init\_timer\_1() {  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);   TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);  TimerLoadSet(TIMER1\_BASE, TIMER\_A, SysCtlClockGet()/10);   IntEnable(INT\_TIMER1A);  TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);   TimerEnable(TIMER1\_BASE, TIMER\_A); }  **const** uint8\_t kodlar[] = {  0b00111111,  0b00000110,  0b01011011,  0b01001111,  0b01100110,  0b01101101,  0b01111101,  0b00000111,  0b01111111,  0b01101111 };  **int** sayi = 0;  **void** timer\_0\_handler() {  // clear time interrupt  TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);   sayi = (sayi + 1) % 10;  GPIO\_PORTB\_DATA\_R = kodlar[sayi]; }  **void** timer\_1\_handler() {  // clear time interrupt  TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);   GPIO\_PORTE\_DATA\_R ^= 0b00010; }   **int** main() {  **volatile** **unsigned** **long** delay;   disable\_interrupts();   // clock source olarak kristali kullan  SysCtlClockSet(SYSCTL\_SYSDIV\_2\_5 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);   init\_timer\_0();  init\_timer\_1();   init\_port\_B();  init\_port\_E();   enable\_interrupts();   GPIO\_PORTB\_DATA\_R = kodlar[sayi];   **while** (1) {  wait\_for\_interrupt();  } } |

## 20150518_131017.jpg

## Uygulama 2:

main.c

|  |
| --- |
| /\*\*  \* ornek ekrani guncellemek icin kullaniliyor.  \*/  #include <stdint.h> #include "inc/lm4f120h5qr.h"  **void** init\_port\_B() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOB;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTB\_DIR\_R |= 0xFF;  GPIO\_PORTB\_AFSEL\_R &= ~0xFF;  GPIO\_PORTB\_DEN\_R |= 0xFF;  }  **void** init\_port\_E() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOE;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTE\_DIR\_R |= 0x0f;  GPIO\_PORTE\_AFSEL\_R &= ~0x0f;  GPIO\_PORTE\_DEN\_R |= 0x0f; }  // 0'dan 9'a kadar olan sayilarin seven segment kodlari // bit sirasi: g f e d c b a uint8\_t kodlar[10] = {  0b0111111,  0b0000110,  0b1011011,  0b1001111,  0b1100110,  0b1101101,  0b1111101,  0b0000111,  0b1111111,  0b1101111 };    **int** main() {  **volatile** **unsigned** **long** delay;   init\_port\_B();  init\_port\_E();   **int** sayi = 1234;   **const** **int** BEKLEME\_LIMIT = 10000;   **while** (1) {  **int** birler = sayi % 10;  GPIO\_PORTE\_DATA\_R |= 0b1111; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[birler];  GPIO\_PORTE\_DATA\_R &= ~0b0001; // birler basamagini aktiflestir  **for** (delay = 0 ; delay < BEKLEME\_LIMIT ; delay++)  /\* bekle \*/;   **int** onlar = (sayi / 10) % 10;  GPIO\_PORTE\_DATA\_R |= 0b1111; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[onlar];  GPIO\_PORTE\_DATA\_R &= ~0b0010; // onlar basamagini aktiflestir  **for** (delay = 0 ; delay < BEKLEME\_LIMIT ; delay++)  /\* bekle \*/;   **int** yuzler = (sayi / 100) % 10;  GPIO\_PORTE\_DATA\_R |= 0b1111; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[yuzler];  GPIO\_PORTE\_DATA\_R &= ~0b0100; // yuzler basamagini aktiflestir  **for** (delay = 0 ; delay < BEKLEME\_LIMIT ; delay++)  /\* bekle \*/;   **int** binler = (sayi / 1000) % 10;  GPIO\_PORTE\_DATA\_R |= 0b1111; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[binler];  GPIO\_PORTE\_DATA\_R &= ~0b1000; // binler basamagini aktiflestir   **for** (delay = 0 ; delay < BEKLEME\_LIMIT ; delay++)  /\* bekle \*/;  } } |



## 4’lu seven segmentte sayıcı

startup\_gcc.c

|  |
| --- |
| **1) include kisimlarindan sonra asagidaki fonksiyon prototiplerini ekleyin**  **extern** **void** timer\_0\_handler();  **extern** **void** timer\_1\_handler();  **2) // The vector table yazan kisimdaki dizide aşağıdaki iki satırda:**  IntDefaultHandler, // Timer 0 subtimer A  IntDefaultHandler, // Timer 1 subtimer A  **IntDefaultHandler yerine ilgili fonksiyonuları yazin**  timer\_0\_handler, // Timer 0 subtimer A  timer\_1\_handler, // Timer 1 subtimer A |

main.c

|  |
| --- |
| /\*\*  \* ornekte systick kesmesi zamani olcmek icin,  \* timer kesmesi ekrani guncellemek icin kullaniliyor.  \*/  #include <stdint.h> #include "inc/lm4f120h5qr.h"  #include "inc/hw\_ints.h" #include "inc/hw\_types.h" #include "inc/hw\_memmap.h" #include "driverlib/sysctl.h" #include "driverlib/timer.h"  **static** **inline** **void** disable\_interrupts() {  asm("CPSID I"); }  **static** **inline** **void** enable\_interrupts() {  asm("CPSIE I"); }  **static** **inline** **void** wait\_for\_interrupt() {  asm("WFI"); }  **void** init\_port\_B() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOB;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTB\_DIR\_R |= 0xFF;  GPIO\_PORTB\_AFSEL\_R &= ~0xFF;  GPIO\_PORTB\_DEN\_R |= 0xFF; }  **void** init\_port\_E() {  **volatile** **unsigned** **long** delay;  SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOE;  delay = SYSCTL\_RCGC2\_R;  GPIO\_PORTE\_DIR\_R |= 0x0f;  GPIO\_PORTE\_AFSEL\_R &= ~0x0f;  GPIO\_PORTE\_DEN\_R |= 0x0f; }  **void** init\_timer\_0(**unsigned** **int** period) {  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);   TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);  TimerLoadSet(TIMER0\_BASE, TIMER\_A, period);   IntEnable(INT\_TIMER0A);  TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);   TimerEnable(TIMER0\_BASE, TIMER\_A); }  **volatile** **int** flag\_ekran\_guncelle = 0;  **void** timer\_0\_handler() {  TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // clear time interrupt   flag\_ekran\_guncelle = 1; }   **void** SysTick\_Init(**unsigned** **long** period) {  NVIC\_ST\_CTRL\_R = 0; // disable SysTick during setup  NVIC\_ST\_RELOAD\_R = period - 1; // reload value  NVIC\_ST\_CURRENT\_R = 0; // any write to current clears it  NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R & 0x00FFFFFF) | 0x40000000; // priority 2  NVIC\_ST\_CTRL\_R = 0x07; // enable SysTick with core clock and interrupts  // enable interrupts after all initialization is finished }  // 1 saniyedeki systick kesmesi sayisi #define SYSTICK\_HZ 100 // baslangictan itibaren olusan systick kesmesi sayisi uint32\_t systick\_count = 0;  **volatile** **int** flag\_sayi\_arttir = 0;  **void** systick\_handler() {  systick\_count++;   // 0.1 saniyede bir sayiyi arttirma flagini 1 yap  **if** ((systick\_count % (SYSTICK\_HZ / 10)) == 0)  flag\_sayi\_arttir = 1; }  // 0'dan 9'a kadar olan sayilarin seven segment kodlari // bit sirasi: g f e d c b a uint8\_t kodlar[10] = {  ***0b0111111***,  ***0b0000110***,  ***0b1011011***,  ***0b1001111***,  ***0b1100110***,  ***0b1101101***,  ***0b1111101***,  ***0b0000111***,  ***0b1111111***,  ***0b1101111*** };  **int** ekran\_guncelle\_no = 0; **void** ekran\_guncelle(**int** sayi) {  ekran\_guncelle\_no = (ekran\_guncelle\_no+1) % 4; // guncellenen ekranin numarasini bir arttir   **if** (ekran\_guncelle\_no == 0) {  **int** birler = sayi % 10;  GPIO\_PORTE\_DATA\_R |= ***0b1111***; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[birler];  GPIO\_PORTE\_DATA\_R &= ~***0b0001***; // birler basamagini aktiflestir  } **else** **if** (ekran\_guncelle\_no == 1) {  **int** onlar = (sayi / 10) % 10;  GPIO\_PORTE\_DATA\_R |= ***0b1111***; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[onlar];  GPIO\_PORTE\_DATA\_R &= ~***0b0010***; // onlar basamagini aktiflestir  } **else** **if** (ekran\_guncelle\_no == 2) {  **int** yuzler = (sayi / 100) % 10;  GPIO\_PORTE\_DATA\_R |= ***0b1111***; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[yuzler];  GPIO\_PORTE\_DATA\_R &= ~***0b0100***; // yuzler basamagini aktiflestir  } **else** **if** (ekran\_guncelle\_no == 3) {  **int** binler = (sayi / 1000) % 10;  GPIO\_PORTE\_DATA\_R |= ***0b1111***; // hepsini kapat  GPIO\_PORTB\_DATA\_R = kodlar[binler];  GPIO\_PORTE\_DATA\_R &= ~***0b1000***; // binler basamagini aktiflestir  } }  **int** main() {  **volatile** **unsigned** **long** delay;   init\_port\_B();  init\_port\_E();  init\_timer\_0(SysCtlClockGet() / 200);  SysTick\_Init(SysCtlClockGet() / SYSTICK\_HZ);   enable\_interrupts();   **int** sayi = 1234;   **while** (1) {  **if** (flag\_ekran\_guncelle == 1) {  flag\_ekran\_guncelle = 0;  ekran\_guncelle(sayi);  }   **if** (flag\_sayi\_arttir == 1) {  flag\_sayi\_arttir = 0;  sayi++;  }   wait\_for\_interrupt();  } } |

