**MSP430 Uyg.28 – ADC Uygulaması\_1**

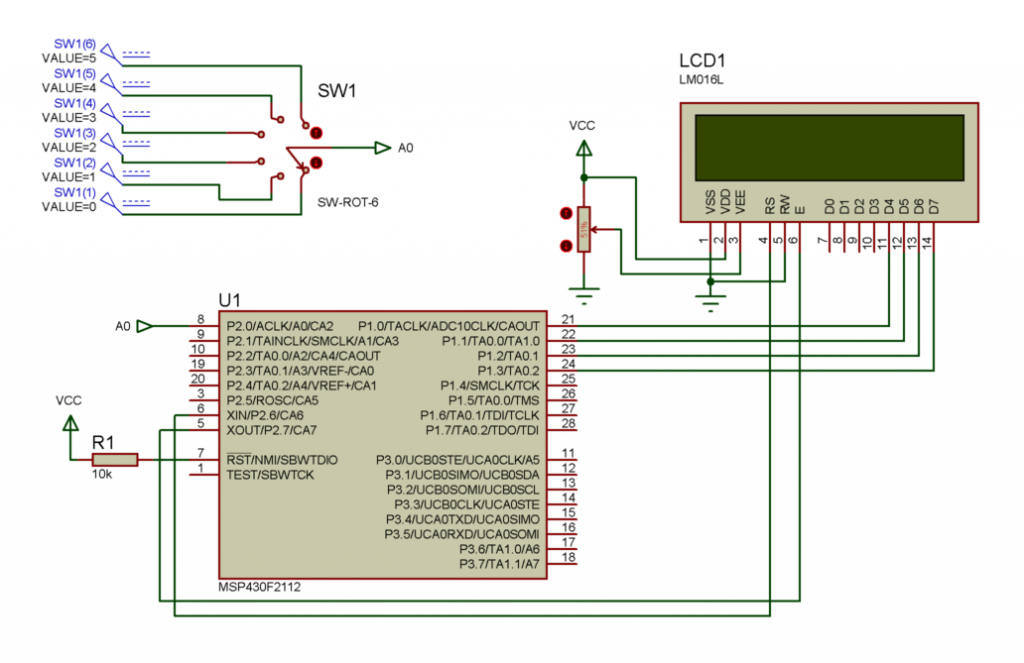
Ferudun GÖKCEGÖZ, 13 Ağustos 2011, Cumartesi



Merhaba arkadaşlar. Yeni bir dersimizle yine sizlerle birlikteyiz. Geçenki derste söylediğimiz gibi bu dersimizde MSP430 ile adc(analog to digital converter) uygulaması yapacağız. Bildiğimiz gibi dış dünyadaki büyüklerin neredeyse tamamı analog büyüklüklerdir. Yani zamanda sürekli olan, genliği değişen büyüklüklerdir. Örneğin; sıcaklık, hız, ivme, basınç, nem vb. Bu gibi fiziksel analog büyüklüklerin ölçülmesinde ADC ler kullanılmaktadır. MSP430 mikrodenetleyicisinde de dahili olarak 10 bitlik bir ADC birimi bulunmaktadır. Bizde bu dersimizde bu birim ile ilgili uygulama yapacağız.

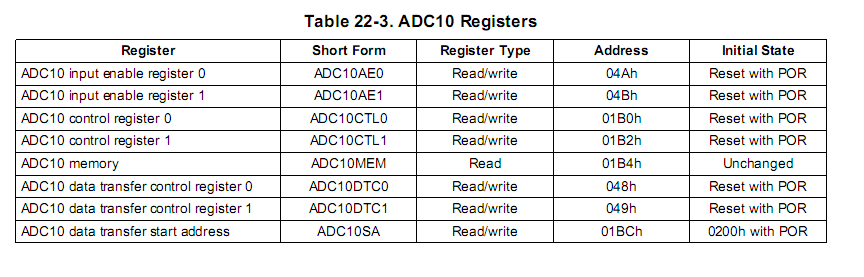
Biz uygulamızı proteus üzerinde yapacağız.Ben proteus üzerindeki msp430 mikrodenetleyici modellerinin adc lerinde problem olduğunu düşünüyorum. Çünkü adc için referans gerilimi bir türlü ayarlanan gibi kabul etmiyor. Sürekli referans gerilim 5V muş gibi davranıyor. Neden olduğunu bir türlü çözemedim. O yüzden proteus taki msp430 ların adc lerinde problem olduğunu düşünüyorum. Bu nedenle referans gerilim her ne kadar 5V olmasada uygulama yaparken 5V olarak düşünelim. Ben uygulamayı launchpad üzerinde denerken bir problem çıkmadı. Normal olarak referans gerilimi Vcc – GND arası kabul ediyor. Ama proteusta olmadı. Hatta bazı modellerde adc hiç çalışmıyor. Bu nedenlede uygulamalarda sürekli kullandığımız MSP430F2232 yerine F2112 modelini kullandım… Baştan bunu söylemek isterim.

Gelelim uygulamaya ait devre şemasına…

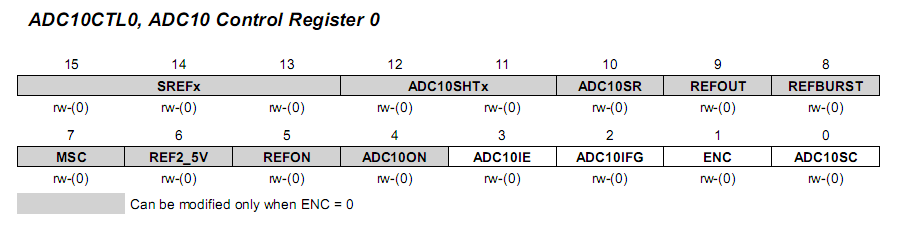


Devrede görüldüğü gibi bir adet 6 kademeli switch bulunmakta. Sırasıyla her bir ucuna, 0V, 1V, 2V, 3V, 4V, 5V luk gerilimler uygulanmaktadır. Switch in diğer ucu ise ADC biriminin A0 kanalına gidiyor. Ve ölçülen değeri görüntülemek için bir adet lcd miz bulunmaktadır. Lcd miz için daha önceki yazılarda oluşturduğumuz kütüphaneleri kullanacağız. Bu yazıda o kısma girmeyeceğiz.

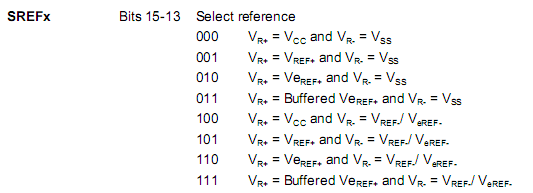
MSP430 un ADC birimine ait register lardan bahsedecek olursak, toplam 8 adet register bulunmaktadır. Sırasıyla, önemli ayarlara biz göz atmak faydalı olacaktır.



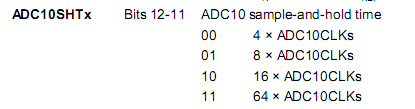
**ADC10CTL0 (ADC10 Control Register 0)**



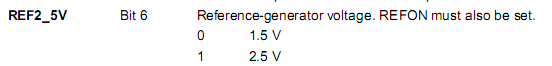
SREFx bitleri referans gerilimi ayarlamak için kullanılırlar.



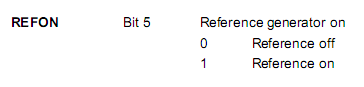
ADC10SHTx bitleri adc örneğinin kaç adc clock darbesinde alınacağını belirler.



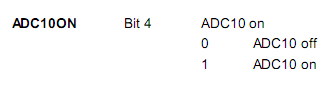
REF2\_5V biti ile de dahili referans kaynağının değeri belirleniyor.



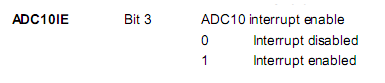
REFON biti ile de dahili referans kaynağı açılıyor.



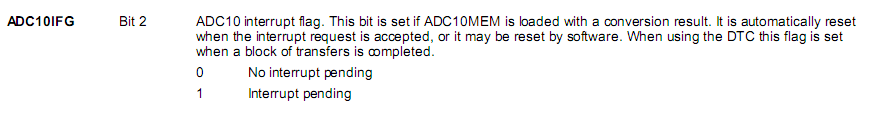
ADC10ON biti ile adc birimi açılıp kapatılıyor.



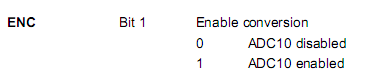
ADC10IE biti ile ise adc çevrimi sonlandığı zaman kesme üretilip üretilmeyeceği ayarlanıyor.



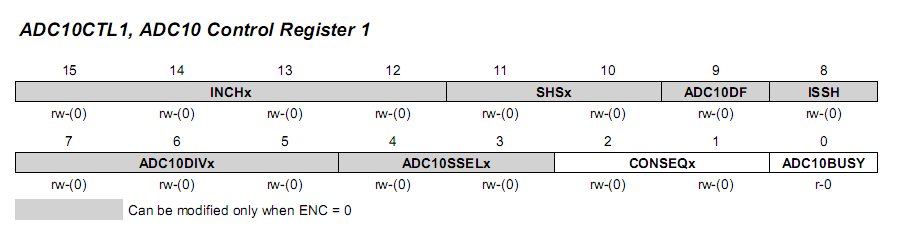
ADC10IFG biti ise adc birimi kesme bayrağıdır. Adc sonucu ADC sonuç register ına yazıldığında bu bayrak set edilir. Kesme aktif ise kesme alt programına dallanılır. Ve bu bit yazılımla temizlenmelidir.



ENC biti ise ADC10SC biti ile birlikte kullanılır. ADC çevrimi tek çevrim modunda ADC10SC biti ile birlikte bu bit te lojik1 yapıldığında başlamaktadır.



**ADC10CTL0 (ADC10 Control Register 1)**



INCHx bitleri ile ADC için çevrim kanalı seçilir.

SHSx bitleri ile de örnekleme işlemi için kaynak seçimi yapılır.

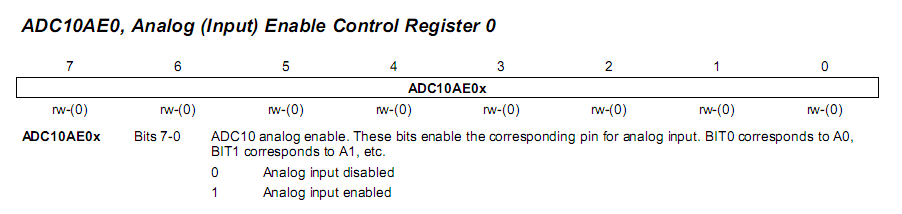
ADC10DF biti ile ise adc sonucunun formatı belirlenir.

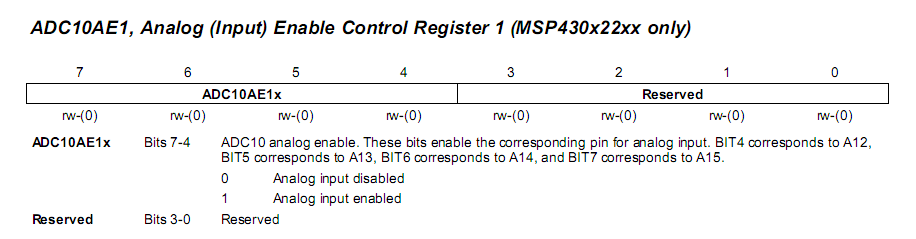
ADC10DIVx bitleri ile de adc clock kaynağının bölücüsü belirlenir.

ADC10SSELx bitleri ile ise clock kaynağı seçilir.

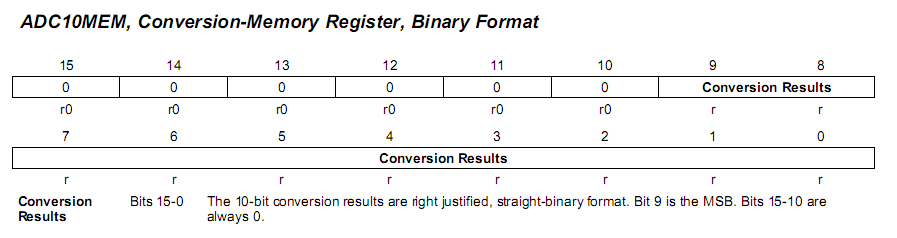
*Bu arada bu iki control register ında gri renkli bitler ENC biti sıfır iken değiştirilebilir Bu yüzden her ihtimale karşı yazılımda adc ayarları yapılırken ENC biti sıfırlanır.*

Diğer register lara da bakacak olursak;





ADC10AE0x ve ADC10AE1x register ları ile ADC girişi aktif edilmek istenen pinler için lojik 1 yapılır. ADCAE1x registerı sadece MSP430x2xx serisi içindir.



ADC10MEM register ı ise 10 bitlik adc sonucunun saklandığı kaydedicidir. ADC birimi için data formatı binary ise 16 bitlik bu register ın ilk 10 biti çevrim sonucunu içerir. Diğer kalan bitler ise 0 olarak okunur. Data formatı “2ye tümleyen” olarak seçilmişse adc çevrim sonucu 2 ye tümlenmiş şekilde sola yatık olarak (left justified) yazılır. Lsb 6 bit ise 0 olarak okunur.

Gelelim uygulamamıza ait yazılımımıza…

|  |  |
| --- | --- |
| **[main.c](http://www.mcu-turkey.com/msp430-uyg-28/" \l "codesyntax_1" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-28/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-28/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#include "io430.h"

#include "in430.h"

#include "lcd\_4bit.h"

unsigned int ADC\_Result = 0x0000;

unsigned char str[4];

void adc\_init(void)

{

ADC10CTL0 & = ~ENC;

ADC10CTL0 = ADC10SHT\_3 + ADC10ON;

ADC10CTL1 = INCH\_0 + ADC10SSEL\_3 + ADC10DIV\_3;

ADC10AE0 = BIT0;

}

void LCD\_Write\_Int (unsigned int val,char mul)

{

char j;

int temp=1;

for(j=0;j<mul;j++)

{

str[j] = (((val/temp)%10)+48);

temp\*=10;

}

for(j=0;j<mul;j++)

lcd\_putch(str[3-j]);

}

void main(void)

{

WDTCTL = WDTPW + WDTHOLD;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

P1SEL = 0x00;

P1OUT = 0x00;

P1DIR = 0xFF;

P2SEL\_bit.P6 = 0;

P2SEL\_bit.P7 = 0;

P2DIR\_bit.P6 = 1;

P2DIR\_bit.P7 = 1;

adc\_init();

lcd\_init();

for(;;)

{

lcd\_goto(1,1);

lcd\_puts(" A0 KANALI ");

ADC10CTL0 |= ADC10SC + ENC;

while(!ADC10CTL0\_bit.ADC10IFG);

ADC\_Result = ADC10MEM;

lcd\_goto(2,4);

LCD\_Write\_Int(ADC\_Result,4);

lcd\_puts("/1023");

\_\_delay\_cycles(10000);

}

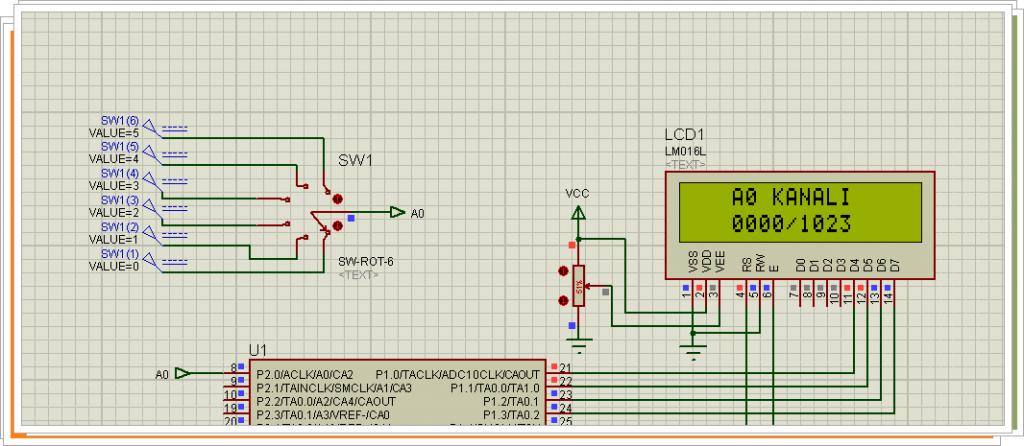
}

Yazılımda ilk olarak “ADC\_Result” isimli integer türden (2 byte) bir değişken tanımlanmıştır. Daha sonra “str” isminde bir unsigned char dizisi tanımlanmıştır. Bu dizi 4 haneli adc sonucu lcd ye yazdırılırken kullanılacaktır. Daha sonra ise adc ayarlarının olduğu adc\_init() isminde bir fonksiyon yazılmış ve ardından da adc sonucunu ekrana yazdırmak için bir fonksiyon yazılmıştır.

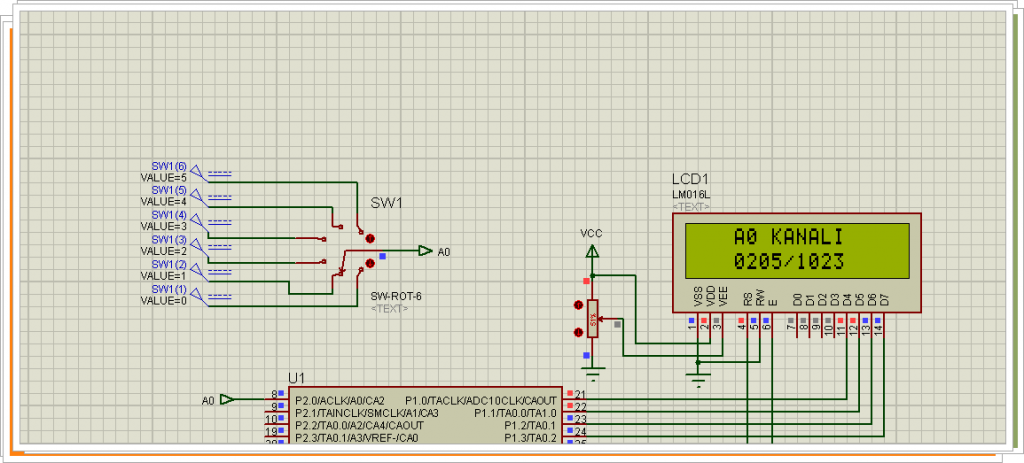
Bu fonksiyon ile main programı içerisinde adc sonucu ekrana yazdırırken kod karmaşası ortadan kalkmıştır. LCD\_Write\_Int() fonksiyonu ile display e 2 byte lık bir integer değere kadar, (65535 e kadar) kaç haneli isteniyorsa o şekilde lcd ye yazdırılabilir. Siz uygulamalarınızda bu fonksiyonu farklı şekillerde dahada genişleterek daha büyük sayılarıda lcd ye yazdırabilirsiniz. Tabi bunun için öncelikle str adlı dizinin boyutunu değiştirmeyi unutmayın.

Main fonksiyonunu incelersek te, ilk olarak WDT,Clock ve Port ayarları yapılmıştır. Ardından adc ve lcd ayarlarının olduğu fonksiyonlar çağrılıp sonsuz döngü içerisine girilmiştir.  Sonsuz döngüde ise ilk olarak lcd displayin birinci satırına “A0 KANALI”  string i yazdırılmış, ardından çevrim başlatılmış ve çevrimin bitmesi beklenmiştir. Bu arada şunuda söyleyelimki, bu uygulamada biz adc kesmesi kullanmadık. Çevrimin bittiğini polling (yoklama) yöntemiyle saptadık. Yazılımda daha sonra ise ADC sonucu ADC10MEM kaydedicisinden okunup lcd ye yazdırmak için kullandığımız fonksiyona gönderilmiştir. Sonrasında bir süre gecikme oluşturulmuştur. Bu işlemler mikrodenetleyicinin enerjisi kesilene kadar bu şekilde süregelmektedir.

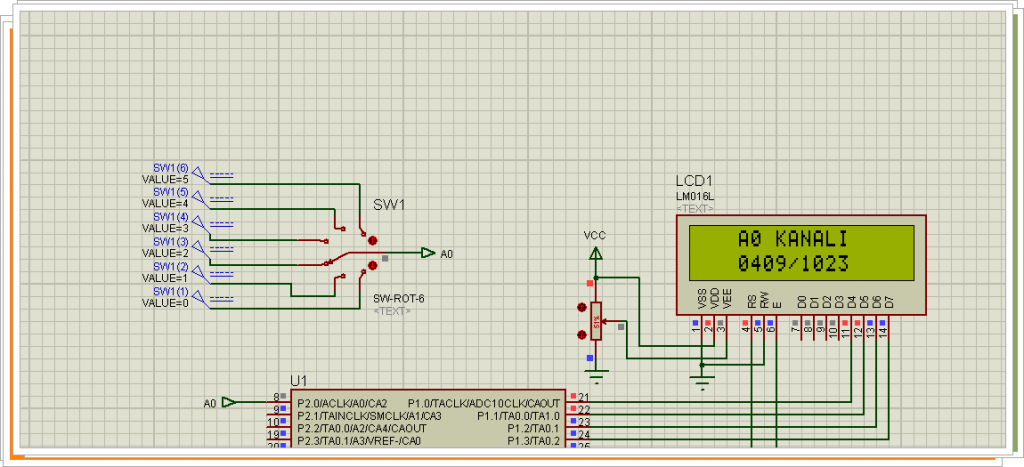
Gelelim simulasyon sonuçlarına… Bu arada adc sonucu 10 bitlik olarak (0~1023/0×0000~0x03FF) arasında displayde gösterilmektedir. Her ne kadar referans gerilimimiz 3.5V civarı olması gereksede, 5V olarak kabul edip o şekilde düşünelim. Proteusun bu probleminin bir çözümünü bulabilirsem daha sonra sizinle paylaşırım :D



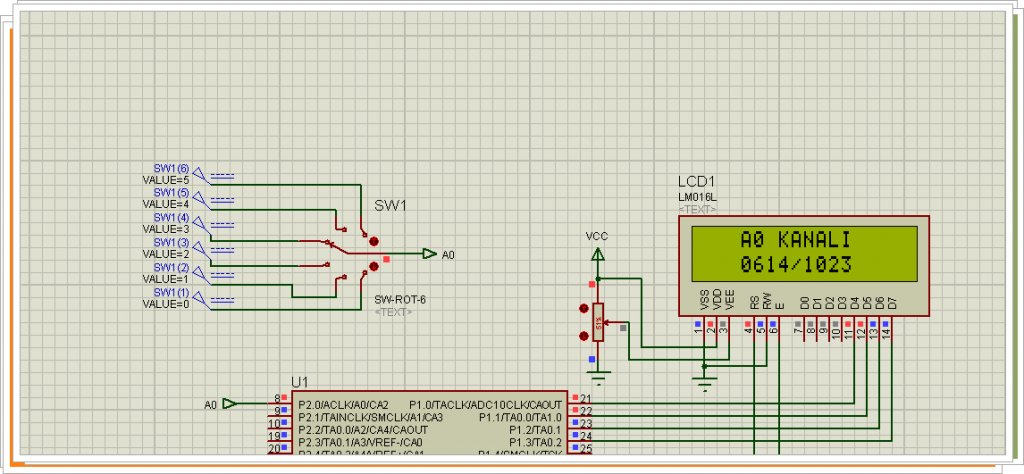
**GERİLİM –> %00**



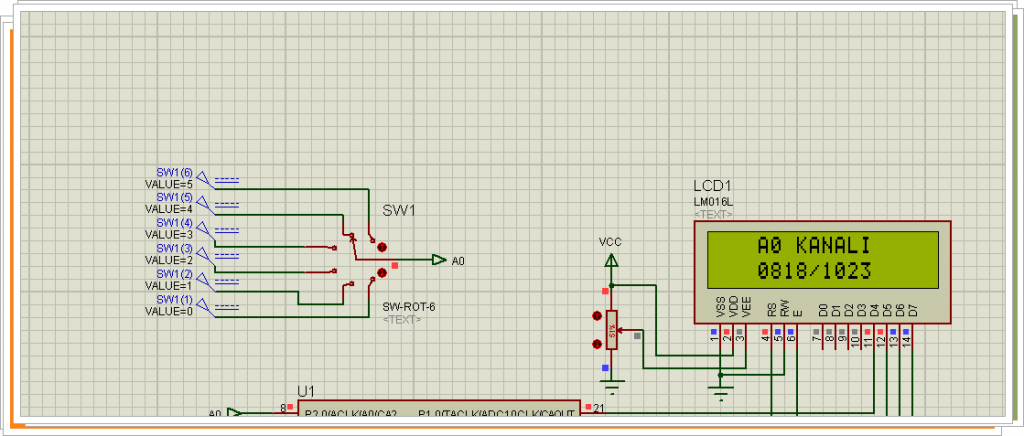
**GERİLİM –> %20**



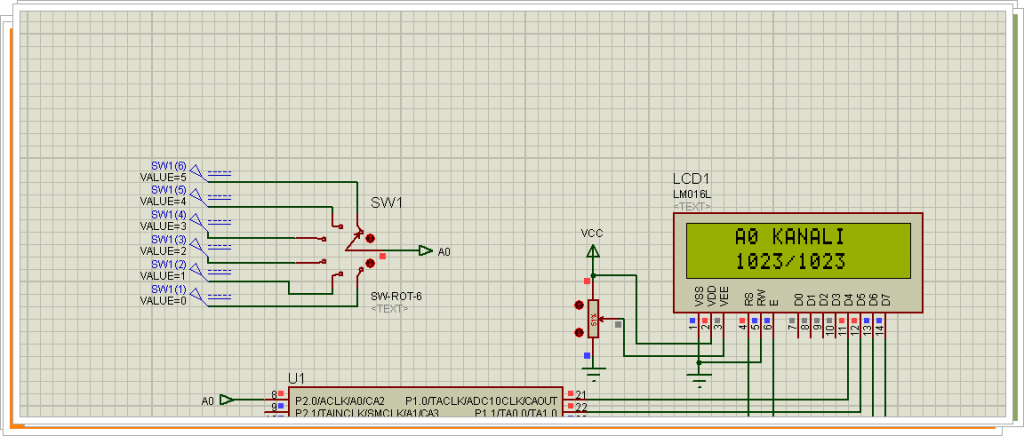
**GERİLİM –> %40**



**GERİLİM –> %60**



**GERİLİM –> %80**



**GERİLİM –> %100**

Geldik bir dersimizin daha sonuna. Siz bu uygulamayı launchpad üzerinde kolaylıkla deneyebilirsiniz. Bir adet potansiyometre size yetecektir. IAR üzerinde lcd sonucunun ADC10MEM kaydedicisinden okunduğu kısımdan sonraki satıra bir *breakpoint* koyarak debug işlemini gerçekleştirebilirsiniz. Benim söyleyeceklerim bu kadar. İnşallah sizlere faydalı olabilmişimdir. Bir dahaki yazımızda yine bir ADC uygulaması yapacağız. Bir dahaki yazımızda görüşmek üzere. Şimdilik Hoşçakalın…

***Ferudun GÖKCEGÖZ***

***fgokcegoz@yahoo.com***