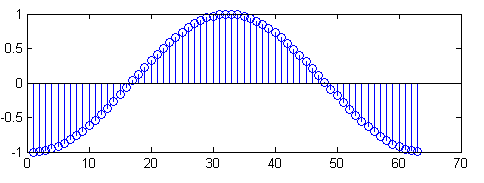
**MSP430 ile Ayrık Zaman Transfer Fonksiyonu Algoritması Koşturma**

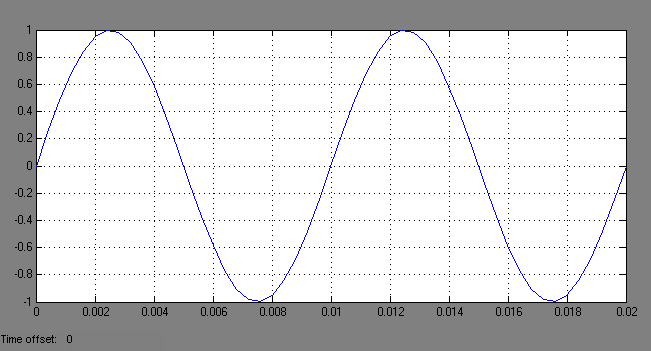
Ferudun GÖKCEGÖZ, 21 Ağustos 2011, Pazar



Merhaba arkadaşlar. Kaldığımız yerden MSP430 uygulamalarına devam ediyoruz. 32.uygulamada söylediğim gibi bu yazımızda Z domenindeki bir transfer fonksiyonunu nasıl mikroişlemci üzerinde koşturacağımızı anlatan bir uygulama yapacağız. Birçoğumuz işaretler ve sistemler dersinden veya otomatik kontrol, dijital kontrol gibi derslerden transfer fonksiyonunun ne olduğunu biliyordur. En genel tanımıyla transfer fonksiyonu, bir sistemin çıkışının girişe oranıdır. H(s), G(s) şeklinde s domeninde(laplace domain) veya H(z), G(z) şeklinde z domeninde(Z domain-ayrık zaman domeni) gösterilirler. s domeni dediğimiz domen, sürekli zaman fonksiyonlar için uygulanan t domenindeki işlem zorluğunu ortadan kaldıran domendir. z domeni ise ayrık zaman sistemlerde kullanılan domendir. Örnekleme zamanına tabidirler ve zamanda sürekli olmayıp, sadece örnek alındığı zamanda bir değere sahiptirler. Bu şekilde z domeninde ifade edilen fonksiyonlar, ayrık şekilde mikroişlemci içerisinde kodlarla koşturulabilirler. Bu yazıdaki uygulama her ne kadar biraz teori içersede, kontrol sistemlerinin aslında bir pratik uygulaması olduğunun göstergesidir.

Biz bu uygulamamızda sinus fonksiyonunu ayrıklaştırıp z domenindeki sinus fonksiyonunu elde edeceğiz. Elde ettiğimiz z domenindeki sinus fonksiyonunu mikroişlemcide kodlarla koşturulabilecek hale getirdikten sonra MSP430 LaunchPad üzerinde kodlarımızı yazıp çıkışa bağlanan DAC (digital analog converter) üzerinden sinus fonksiyonumuzu göreceğiz.

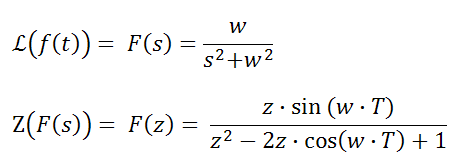
Örnek olarak bir sinus fonksiyonu aşağıdaki şekilde olmaktadır.



Genel bir ifade ile sürekli zamanda sinüs fonksiyonu ;

http://www.mcu-turkey.com/wp-content/uploads/2011/08/0516.png

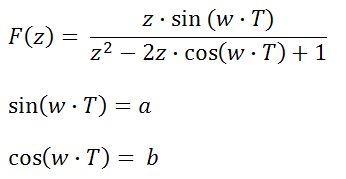
şeklinde olmaktadır. Burada f : frekans t ise zamandır. Yukarıdaki sinüs grafiği ise f=100 Hz lik bir sinüs dalgasıdır. Görüldüğü gibi yukarıdaki fonksiyon sürekli – zaman bir fonksiyondur ve mikroişlemcide koşturulabilmesi için ayrık – zaman forma getirilmesi gerekir. Bu nedenle fonksiyonumuzu ilk olarak s domeninde sonra ise z domeninde(ayrık zaman formunda) yazacak olursak…



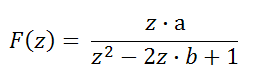
Burada ‘T’ örnekleme zamanını temsil etmektedir.  Bu şekilde sinus fonksiyonumuzun ayrık zaman formunu elde etmiş olduk. Bu aşamadan sonra elde ettiğimiz ayrık zaman formundaki denklemi programlama yöntemlerinden biriyle programlanabilir hale getirmek gerekir.

Dijital kontrol sistemlerinde kullanılan birkaç tane programlama yöntemleri bulunmaktadır. Seri, paralel, doğrudan programlama yöntemi gibi. Biz ayrık zaman formundaki fonksiyonu programlamak için doğrudan programlama yöntemini kullanacağız.

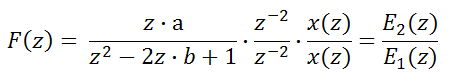
Daha önceden sinüs fonksiyonunun ayrık zaman formunu elde etmiştik.



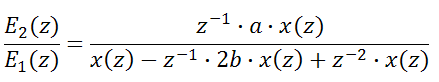
olarak a ve b ifadelerinide F(z)’e ilave edecek olursak denklemimiz yeni hali aşağıdaki şekilde olacaktır.



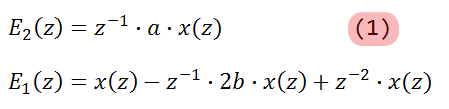
Doğrudan programlama yönteminde, Z domenindeki denklemimizin pay ve paydası karakteristik denklemin en yüksek dereceli teriminin negatifi ile ve x(z) değeri ile çarpılmakta ve sonuç aşağıdaki gibi matematiksel olarak ifade edilmektedir.



Denklemimizin pay ve paydasını aynı sayı ile çarptığımızdan dolayı değeri değişmemektedir. Bu haliyle denklemin son hali,



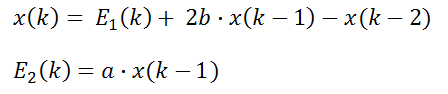
Pay ve paydayı ayrı olarak yazarsak,



Olarak elde edilecektir. Bu aşamadan sonra E1e ait denklem tekrar düzenlenip, x(z) çekilirse aşağıdaki şekilde elde edilecektir.

http://www.mcu-turkey.com/wp-content/uploads/2011/08/126.png

(1) ve (2) denklemlerini son olarak aşağıdaki gibi sürekli zamanda yazabiliriz.



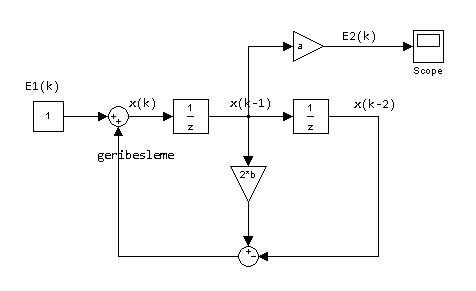
x(k) : durum değişkeninin şu anki değeri

x(k-1) : durum değişkeninin bir önceki değeri

x(k-2) : durum değişkeninin iki önceki değeri

Burada şu soru akla gelebilir. Ayrık zaman formdan sürekli zaman forma geçtik ama hala denklemlerimiz t(zaman) ye bağlı değil. Şöyle cevap verebilirim ki, ayrık zaman formdan sürekli zaman forma geçerken ifadeler zamana bağlı olarak oluşmaz. Ayrık zamandaki fonksiyonlar sadece örnekleme anlarında tanımlı oldukları için, sürekli zamana dönüştüklerinde de sadece ‘k’ anlarında tanımlı olurlar.

Bu aşamadan sonra denklerimiz son halini almaktadır. Ve sinüs fonksiyonuna ait blok diyagramını matlab/simulink ortamında çizecek olursak,



Yukarıdaki blok diyagramında gördüğümüz gibi sinüs fonksiyonunun ayrık zaman blok diyagramını oluşturduk. Blok diyagramındaki 1/z ifadeleri birim gecikmelerdir. Z domeninde 1/z olarak ifade edilirler. Yukarıdaki blok diyagrama göre sembolik olarak bir programlama mantığı oluşturmak istersek şu şekilde yazabiliriz.

1) Giriş =1; Çıkış=0; Geribesleme=0; Gecikme\_1=0; Gecikme\_2=0;

2) Çıkış = Gecikme\_1 \* a;

3) Geribesleme = Gecikme\_1\*2\*b – Gecikme\_2;

4) X\_k = Giriş + Geribesleme;

5) Gecikme\_2 = Gecikme\_1;

6 ) Gecikme\_1  = X\_k;

7) 2.satıra dön.

Bu aşamadan sonra yapacağımız tek şey oluşturduğumuz sembolik programlama mantığını mikrodenetleyiciye uyarlayıp, periyodik olarak yani örnekleme zamanına tabi olarak koşturmak olacaktır. Benim bu iş için MSP430 LaunchPad üzerinde denemek için yazdığım kodu paylaşayım.

|  |  |
| --- | --- |
| **[main.c](http://www.mcu-turkey.com/msp430-ile-ayrik-zaman-transfer-fonksiyonu-algoritmasi-kosturma/" \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-ile-ayrik-zaman-transfer-fonksiyonu-algoritmasi-kosturma/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-ile-ayrik-zaman-transfer-fonksiyonu-algoritmasi-kosturma/#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"

int Input=1;

int Output=0;

float FeedBack=0;

float x\_k=0;

float x\_k\_1=0;

float x\_k\_2=0;

float a=0.0628;

float b=0.9980;

float temp;

#define Clock P1OUT\_bit.P0

#define SS P1OUT\_bit.P1

#define Data P1OUT\_bit.P2

#define Clock\_DIR P1DIR\_bit.P0

#define SS\_DIR P1DIR\_bit.P1

#define Data\_DIR P1DIR\_bit.P2

void MCU\_init (void)

{

WDTCTL = WDTPW + WDTHOLD;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

TACCTL0=CCIE;

TACCR0=10000;

TACTL=TASSEL\_2 + MC\_1 + TAIE;

SS\_DIR = 1;

Clock\_DIR = 1;

Data\_DIR = 1;

SS = 1;

Clock = 1;

Data = 1;

 }

void DAC\_Output(unsigned int valueDAC)

{

unsigned char j=0;

SS = 0;

valueDAC |= 0x3000;

for(j=0; j<16; j++)

{

Clock = 0;

Data = ((valueDAC>>(15-j)) & 0x0001);

Clock = 1;

\_\_delay\_cycles(4);

Clock = 0;

}

SS = 1;

}

void main(void)

{

MCU\_init();

DAC\_Output(0);

\_BIS\_SR(GIE);

for(;;);

}

#pragma vector=TIMERA0\_VECTOR

\_\_interrupt void Timer\_A0 (void)

{

temp = x\_k\_1\*a\*130;

Output = (int)temp;

DAC\_Output(Output);

FeedBack = (x\_k\_1\*2\*b)-(x\_k\_2);

x\_k = Input+FeedBack;

x\_k\_2 = x\_k\_1;

x\_k\_1 = x\_k;

TACTL\_bit.TAIFG = 0;

}

Kodumuzundan bahsedecek olursak, ilk olarak main fonksiyonunda ilk ayarların bulunduğu MCU\_init() altprogramı çağrılmıştır. Ardından da DAC  a sıfır değeri gönderilmiştir. Daha sonra ise tüm kesmeler açılmış ve sonsuz döngüye girilmiştir. Bu aşamada timer kesmesi beklenmektedir. Timer ımız 10 msn ye ayarlanmıştır. Yani 10 msn de bir timer kesme alt programına gidilecek içerdeki kodlar icra edilecektir. Bu tam olarak şu anlama gelmektedir ki, 10 msn örnekleme zamanına bağlı olarak, timer kesme alt programındaki kodlar yani sinüs fonksiyonuna ait ayrık-zaman algoritmasının kodları koşturulacaktır. Devrede kullandığım DAC, microchip firmasına ait MCP4921 kodlu DAC entegresidir. SPI dan haberleşebilir. Yalnız LaunchPad üzerinde tüm denemelerime rağmen bu entegre ile SPI dan haberleşemediğim için bu DAC\_Output() adındaki fonksiyonla DAC a veri göndermeyi tercih ettim. Ben  bu uygulamayı daha önce Analog Devices firmasına ait ADuC841 model 8052(8051) tabanlı bir mikrodenetleyicide yapmıştım. O mikrodenetleyicide dahili olarak 12 bit lik DAC bulunuyordu. O yüzden harici olarak DAC a ihtiyaç duymamıştık.

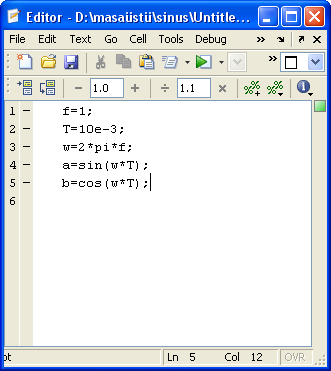
Yazının ilk kısımlarında “a” ve “b” olarak iki adet değişkenimiz vardı. Biz bu sinüslü ve kosinüslü ifadeleri kısaltmak için kullanmıştık. Yazılımda da ilk olarak değişken tanımlamaları yapılırken, bu değerler float olarak girilmiştir. Bunu şu şekilde ifade edebilirimki, “a” ve “b” değerleri yani sin(w\*T) ve cos(w\*T) değerlerindeki en önemli kısım “w” dır. Yani açısal hızdır. Siz bu a ve b değerleri ile üretmek istediğiniz sinüs dalgasının frekansını belirlemiş oluyorsunuz.

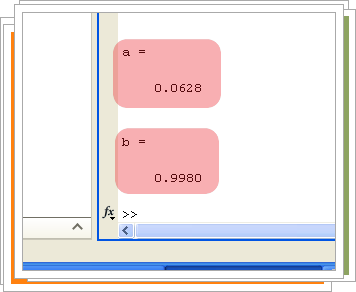
Hepimiz biliyoruzki, w = 2\* pi \* f  tir.  “f”  değerini uygun olarak seçtiğimizde, a ve b değerleride otomatik olarak elde edilecektir. Yazılımda görülüyorki, a ve b değerleri;

a=0.0628;

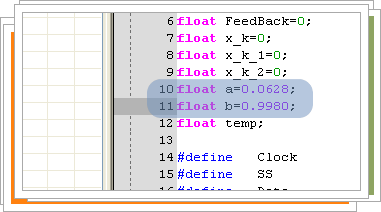
b=0.9980;

Bu değerleri hesaplamak için Matlab te ufak bir .m file hazırladım. Bunuda sizlerle paylaşayım.

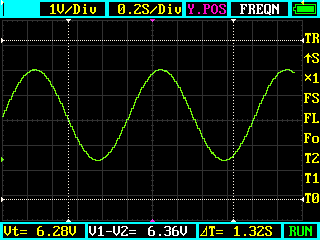




Yukarıdaki .m file dan görüldüğü gibi üretmek istediğimiz sinüs sinyalinin kolaylık olması açısından 1Hz olarak seçmişiz. Örnekleme periyodunu ise T=10 msn olarak belirlemişiz. Buna göre bir alttaki resimde görülüyorki, a ve b değerleri bizim için hesaplanmış. Artık bu değerleri yazılımda kullanabiliriz.



Uygulamaya ait osiloskop görüntüsünü ve videosunu da verelim.



Görüldüğü gibi başarılı bir şekilde sinüs sinyali elde edilmiştir. Teorik olarak istenilen frekanslarda bir sinyal elde edilmesi mümkün görünüyorken, pratikte mümkün değildir. Çünkü bizi kısıtlayan bazı unsurlar vardır. Bunlardan birisi işlemcinin frekansıdır. Koşturulan alt program içerisinde float değişkenler üzerinden işlemler yapılmaktadır. Bu nedenle bu işlemler uzun sürmektedir. Üretmek istediğimiz sinüs fonksiyonunun frekansı ne kadar büyürse, örnekleme hızımızında bir o kadar büyük olması gerekir. Buda bizim timer kesme alt programımızdaki kodların çok kısa süre içerisinde icra edilmesini gerektirir. Bu nedenle kullandığınız mikrodenetleyiciye bağlı olarak sınırlarınız bulunmaktadır.

Ayrıca şunu da söylemeliyimki, bu konu her ne kadar biraz teori içerse de, kontrol sistemlerinin uygulaması açısından çok önemlidir. Şöyle örnek verebiliriz. Mesela bir P,PI,PD veya PID kontrolör tasarladınız. Ve tasarladığınız PID kontrolör için Kp,Ki ve Kd değerlerinizde belli. Bu değerlere bağlı olarak ayrık – zamanda bir transfer fonksiyonunuzda var. Peki bunu nasıl mikrodenetleyicide veya dsp de koşturacağım konusuna girildiğinde bu uygulama temel oluşturacaktır. Çünkü ayrık zamanlı bir sistem mikrodenetleyicide koşturulacak ise bu yazıdaki aşamalardan geçmek zorundadır.

Vakit ayırıp yazıyı okuyan arkadaşlarıma şimdiden teşekkür ederim. İnşallah faydalı olabilmişimdir. Bu yazı biraz uzun ve teorik bir yazı olsada, dikkatle okunduğunda rahatlıkla anlaşılacağını düşünüyorum. Bir dahaki yazımızda görüşmek üzere. Hoşçakalın…

***Ferudun GÖKCEGÖZ***

***fgokcegoz@yahoo.com***