

پروژه درس ریزپردازنده و زبان اسمبلی

طاها موسوى 98243058

97243063 جم

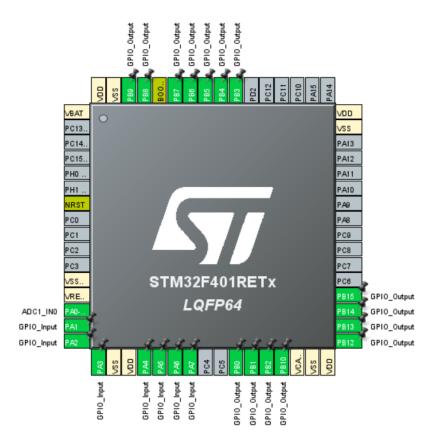
گروه 2

مقدمه

در این پروژه قصد داریم با مبدل آنالوگ به دیجیتال، سیگنال DMTF را دیکود کرده و مقدار آن را روی سون سگمنت نمایش دهیم. سیگنال های DMTF از Or کردن سیگنال های سینوسی به دست می آیند.

تنظیمات cubemx

طرح کلی میکرو:



از gpio output ها (ینی pb0 تا pb6 و pb7 تا pb15) برای وصل شدن به سون سگمنت و نمایش نتیجه بر روی آن ها استفاده شده است.

ADC1 Mode and Configuration						
Mode						
✓ INO						
■ IN1						
■ IN2						
■ IN3						
■ IN4						
■ IN5						
■ IN6						
Configuration						
Reset Configuration						
	A Settings Settings					
Configure the below parameters :						
Q Search (Ctrl+F)	0					
∨ ADCs_Common_Settings						
Mode	Independent mode					
∨ ADC_Settings						
Clock Prescaler	PCLK2 divided by 2					
Resolution	12 bits (15 ADC Clock cycles)					
Data Alignment	Right alignment					
Scan Conversion Mode	Enabled					
Continuous Conversion Mode	Disabled					
Discontinuous Conversion M Disabled						
DMA Continuous Requests Disabled						
End Of Conversion Selection EOC flag at the end of single channel c						
∨ ADC Regular ConversionMode						

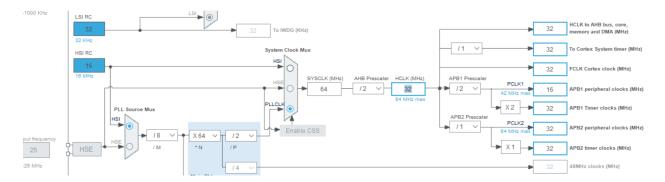
در اینجا adc0 را فعال کرده ایم که به pa0 وصل میباشد.

جواب را در 12 بیت میریزیم.

Scan conversion را enable می کنیم.

سمپلینگ تایم برابر 3 کلاک میباشد. به این معنی که 3 کلاک طول میکشد تا نمونه برداری انجام شود.

Number of conversions را برابر 1 گذاشتیم یعنی تنها یک چنل از اینجا انتخاب کردیم.



با توجه به تنظیم prescaler این مقدار تقسیم به 2 میشود.

از این مقادیر، میخواهیم Sampling rate را به دست آوریم. برای به دست آوردن Sampling rate را به دست آوردن Sampling rate

(APB2Clk/Prs)/(resolution + samplingtime)

برای به دست آوردن این فرمول از

https://community.st.com/s/question/0D50X00009XkfjOSAR/adc-sampling-rate-calculation

کمک گرفتیم. بنابراین محاسبات Sampling rate برای خودمان چنین است:

$$\frac{32M/2}{12+3} = 34944$$

از لحاظ تئوری، مقدار به دست آمده به این شکل است. اما از لحاظ عملی، ممکن است به این شکل نباشد.

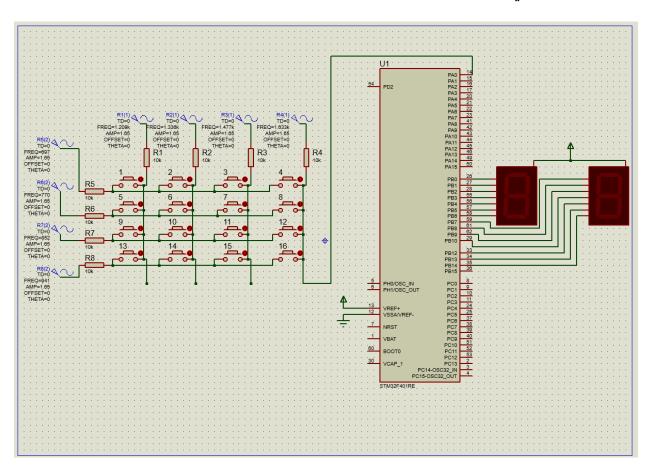
	DMA Settings			GPIO Settings	
	Parameter Settings		User Constants		
NVIC Interrupt Table		Enabled	Preem	ption Priority	Sub Priority
ADC1 global interrupt			0		0

اینتراپت را فعال نکردیم یعنی از روش polling استفاده میکنیم.

اجزا و تنظيمات پروتئوس

از اجزای زیر برای شبیه سازی در پروتئوس استفاده کردیم:

- Stm32f401re microcontroller -
 - مقاومت
 - دو عدد 7-segment دوتایی
 - ژنراتورهای تولید سیگنال
 - 16 عدد كليد



همانطور که مشاهده میشود 8 تا ژنراتور تولید سیگنال هرکدام به یک مقاومت وصل شده اند و هر کلید دو تا دو تا آن ها را با هم O۲ میکند.

توضیحات مربوط به کد و راه حل

برای محاسبه سیگنال از الگوریتم goertzel استفاده کردیم که طبق موارد بیان شده در صورت پروژه عمل میکند.

```
Q_0 = (coef_k \times Q_1[n]) - Q_2[n] + x[n], Q_1 = Q_0[n-1], Q_2 = Q_1[n-1], X[n] = Q_0[n-1], X[n] = U[n-1] نمونه خوانده شده از مبدل آنالوگ به دیجیتال [k = \lceil 0.5 + ((N \times pare + pare
```

 Q_0 در معادلات فوق با فرض این که آرایه x به طول N نمونه از سیگنال ورودی نمونهبرداری شده باشد، x[n] و رودی فعلی، x[n] خروجی فعلی و x[n] و x[n] خروجیهای محاسبات مرحله قبل هستند. با توجه به مقادیر فوق، توان شکل موج ورودی برای فرکانس مورد جستجو در هر نوبت محاسبه عبارتست از

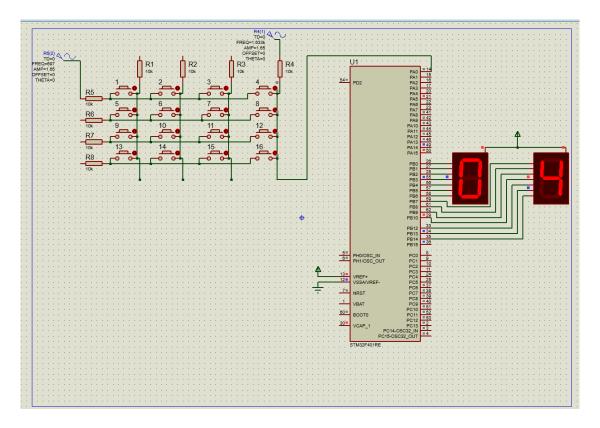
```
دامنه = توان ^2 = Q_1^2[N] + Q_2^2[N] - (coef_K \times Q_1[N] \times Q_2[N])
```

```
int
float
       floatnumSamples;
       omega, sine, cosine, coeff, q0, q1, q2, magnitude, real, imag;
float scalingFactor = numSamples / 2.0;
floatnumSamples = (float) numSamples;
k = (int) (0.5 + ((floatnumSamples * TARGET_FREQUENCY) / SAMPLING_RATE));
omega = (2.0 * 3.1415* k) / floatnumSamples;
sine = sin(omega);
cosine = cos(omega);
coeff = 2.0 * cosine;
q0=0;
\alpha 1 = 0:
q2=0;
for(i=0; i<numSamples; i++)</pre>
    q2 = q1;
    q1 = q0;
    q0 = coeff * q1 - q2 + adc[i];
// calculate the real and imaginary results
// scaling appropriately
real = (q0 - q1 * cosine) / scalingFactor;
imag = (-ql * sine) / scalingFactor;
magnitude = sqrtf(real*real + imag*imag);
return magnitude;
```

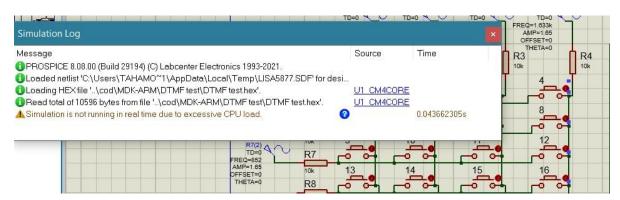
```
int main (void) {
    HAL Init();
    SystemClock Config(); /* Configure the system clock */
    MX GPIO Init();
    MX ADC1 Init();
    showOn7Segment(OXFFFF);
    hadcl.Init.NbrOfConversion=1;
    HAL ADC Init (shadel);
int numberOfShift = 7:
    while (1) {
     for(i=0;i<1400;i++){
          HAL ADC Start(shadcl);
          if (HAL_ADC_PollForConversion(shadcl, 1) == HAL_OK)
                     adc[i]=HAL ADC GetValue(shadcl);
          HAL_ADC_Stop(shadcl);
          1:
            if((goertzel_mag(1400,697,25000))>80.0){
               if((goertzel_mag(1400,1209,25000))>80.0)
                             showOn7Segment((seg[1/10])+((seg[1])<< numberOfShift));</pre>
               if((goertzel_mag(1400,1336,25000))>80.0)
                             showOn7Segment((seg[2/10])+((seg[2])<< numberOfShift));
               if((goertzel mag(1400,1477,25000))>80.0)
                           showOn7Segment((seg[3/10])+((seg[3])<< numberOfShift));</pre>
               if((goertzel_mag(1400,1633,25000))>80.0)
                               showOn7Segment((seg[4/10])+((seg[4])<< numberOfShift));
          i=0:
          if((goertzel mag(1400,770,25000))>80.0){
             if((goertzel_mag(1400,1209,25000))>80.0)
                        showOn7Segment((seg[5/10])+((seg[5])<< numberOfShift));
             if((goertzel_mag(1400,1336,25000))>80.0)
                        showOn7Segment((seg[6/10])+((seg[6])<< numberOfShift));
             if((goertzel_mag(1400,1477,25000))>80.0)
                        showOn7Segment((seg[7/10])+((seg[7])<< numberOfShift));</pre>
             if((goertzel_mag(1400,1633,25000))>80.0)
                      showOn7Segment((seg[8/10])+((seg[8])<< numberOfShift));</pre>
             1:
          i=0:
          if((goertzel_mag(1400,852,25000))>80.0){
               if((goertzel_mag(1400,1209,25000))>80.0)
                            showOn7Segment((seg[9/10])+((seg[9])<< numberOfShift));
               if((goertzel_mag(1400,1336,25000))>80.0)
                            showOn7Segment((seg[10/10])+((seg[0])<< numberOfShift));
               if((goertzel_mag(1400,1477,25000))>80.0)
                            showOn7Segment((seg[11/10])+((seg[1])<< numberOfShift));
               if((goertzel_mag(1400,1633,25000))>80.0)
                            showOn7Segment((seg[12/10])+((seg[2])<< numberOfShift));
          i=0;
          if((goertzel_mag(1400,941,25000))>80.0){
               if((goertzel_mag(1400,1209,25000))>80.0)
                            showOn7Segment((seg[13/10])+((seg[3])<< numberOfShift));
               if((goertzel mag(1400,1336,25000))>80.0)
                          showOn7Segment((seg[14/10])+((seg[4])<< numberOfShift));
               if((goertzel mag(1400,1477,25000))>80.0)
                          showOn7Segment((seg[15/10])+((seg[5])<< numberOfShift));</pre>
               if((goertzel mag(1400,1633,25000))>80.0)
                          showOn7Segment((seg[16/10])+((seg[6])<< numberOfShift));</pre>
               1:
```

در تابع مین با بررسی سطر و ستون متوجه میشویم که کدام کلید فشرده شده و نتیجه مربوط به آن را با تابع showOn7Segment روی سون سگمنت نمایش میدهیم.

مشاهده خروجی پروژه در پروتئوس

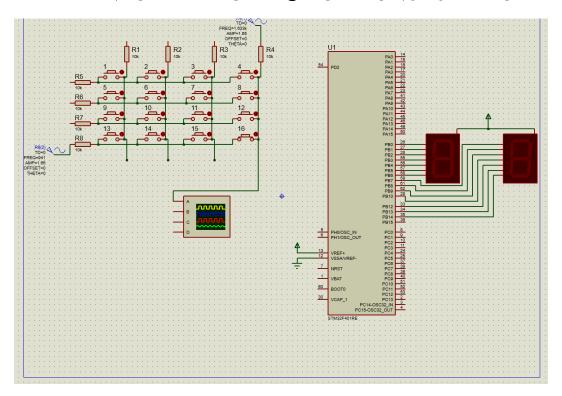


با کلیک بر روی هر دکمه، عدد معادل آن دکمه باید روی سون سگمنت نمایش داده شود. اما اینجا با یک چالش رورو شدیم.

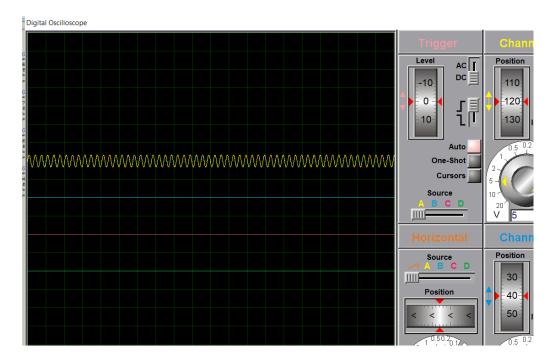


همانطور که مشاهده میشود. شبیه سازی بصورت real time اجرا نمیشود. به همین صورت سرعت شبیه سازی پس از کلید بر روی دکمه و Or شدن سیگنال ها به شدت پایین میآید. به طوری که ممکن است چندین دقیقه طول بکشد تا نتیجه Or شدن سیگنال ها به دست آید و عدد روی سون سگمنت نمایش داده شود.

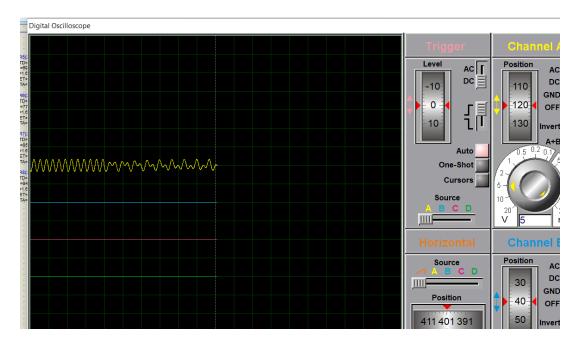
برای اینکه در این زمان مطمعن شویم که فرایند متوقف شده و انجام نمیشود یا به درستی درحال انجام هست از یک اسیلوسکوپ برای دیدن نتایج سیگنال ها استفاده کردیم.



قبل از کلیک دکمه:



بعد از کلیک دکمه:



بخاطر این چالش گرفتن خروجی ها و نمایش آن ها به شدت برای ما سخت شد. موفق شدیم چند تا خروجی رو پشت سر هم بگیریم که بصورت ویدیو در فایل پروژه پیوست شده است.