دانشگاه صنعتی امیرکبیر

دانشکده مهندسی پزشکی

گزارش آزمایشگاه میکروپروسسور

آزمایش ۶: مبدل آنالوگ به دیجیتال – مبدل دیجیتال به آنالوگ

گروه ۲:

حمیدرضا ابوئی مهریزی

Click here to enter text.

تاریخ اتمام آزمایش: ۱۴۰۱/۰۴/۱۷

آن چه از این آزمایش فرا گرفته‌اید:

مقدمه

پرسش‌ها و آزمایش‌ها

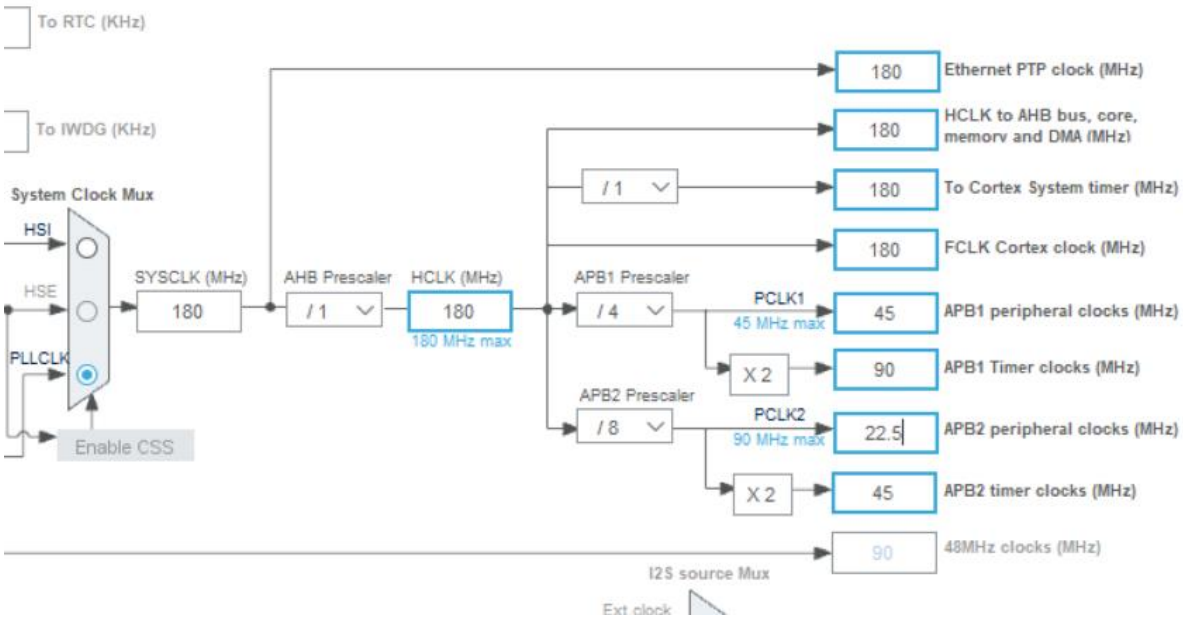
بخش اول

؟ سوأل ۱:

حداکثر رزولوشن مبدل آنالوگ به دیجیتال در خانواده STM32F4xx برابر ۱۲ بیت است

؟ سوأل ۲:

حداکثر فرکانس کاری ADC همانطور که در دیتاشیت ذکر شده برابر با ۳۶ مگاهرتز است.



می‌توانیم از APB2 Prescaler برای تقسیم فرکانس استفاده کنیم.

بنابراین با توجه به مقادیر این تقسیم کننده فرکانس، مقادیر تقسیم ۸ و ۱۶ در محدوده مجاز قرار می‌گیرند.

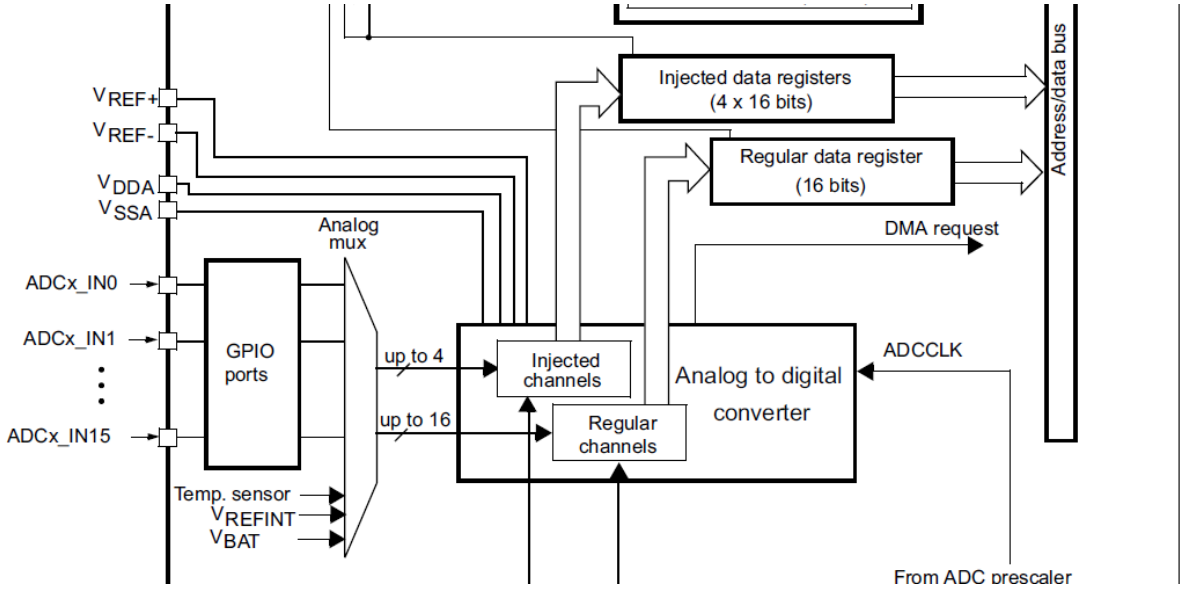
زمان تبدیل از فرمول زیر محاسبه میشود:

$$T_{conv=}SamplingTime + 12cycles$$

از آنجاییکه کمترین زمان برای زمان نمونه برداری سه سیکل است. پس حداقل زمان تبدیل ۱۵ سیکل است. حداقل زمان تبدیل به ازای فرکانس ۳۰ مگاهرتز و ۱۲ بیت ، نیم میکرو ثانیه است

البته در رزولوشن های کمتر، تعداد سیکل‌های مورد نیاز برای کانورژن کمتر است

؟ سوأل ۳:



Vref+و Vref-

Table 65. ADC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, 1.8 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	Input, analog supply	Analog power supply equal to V_{DD} and 2.4 V ≤ V_{DDA} ≤ V_{DD} (3.6 V) for full speed 1.8 V ≤ V_{DDA} ≤ V_{DD} (3.6 V) for reduced speed
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
V_{SSA}	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_IN[15:0]	Analog input signals	16 analog input channels

؟ سوأل ۴:

مبدل دیجیتال به آنالوگ خانواده stm32f4xx می‌تواند ۸ یا ۱۲ بیتی باشد.

❓ سوأل ۵:

خیر ، بعد از یک مدت زمان t wakeup فعال و قابل استفاده است. مقدار آن تیپیکال ۶.۵ و حداکثر ۱۰ میکروثانیه است.

❓ سوأل ۶:

$$Vout = \left(\frac{RL}{Rout + RL}\right) * V_{DAC}$$

$$error = \frac{V_{DAC} - Vout}{V_{DAC}} < 0.05$$

$$1 - \frac{RL}{Rout + RL} < \frac{5}{100} \quad \rightarrow \quad \frac{Rout}{Rout + RL} < \frac{5}{100}$$

$$20 * 15k < 15k + RL$$

$$RL > 285k$$

حداقل مقاومت باید از ۲۸۵ کیلو بیشتر باشد، در صورت استفاده از مقاومت خروجی کمتر باید از یک طبقه تقویت کننده استفاده کرد یا از بافر خروجی DAC استفاده کرد.

❓ سوأل ۷:

همانطور که در قسمت قبل گفته شد می‌توان بافر خروجی را فعال کرد.

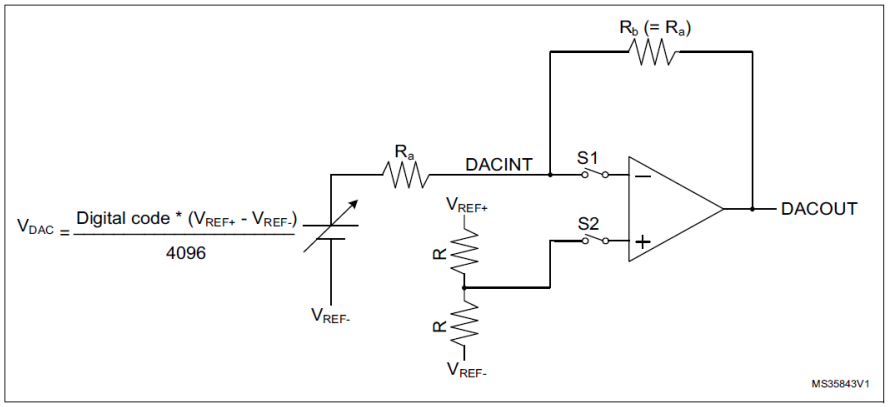
$$V_{DAC} = \frac{digital\ code * \left(V_{reff+} - V_{reff-}\right)}{4096}$$

$$V_{S2} \models V_{S1} = \frac{\left(V_{reff+} - V_{reff-}\right)}{2}$$

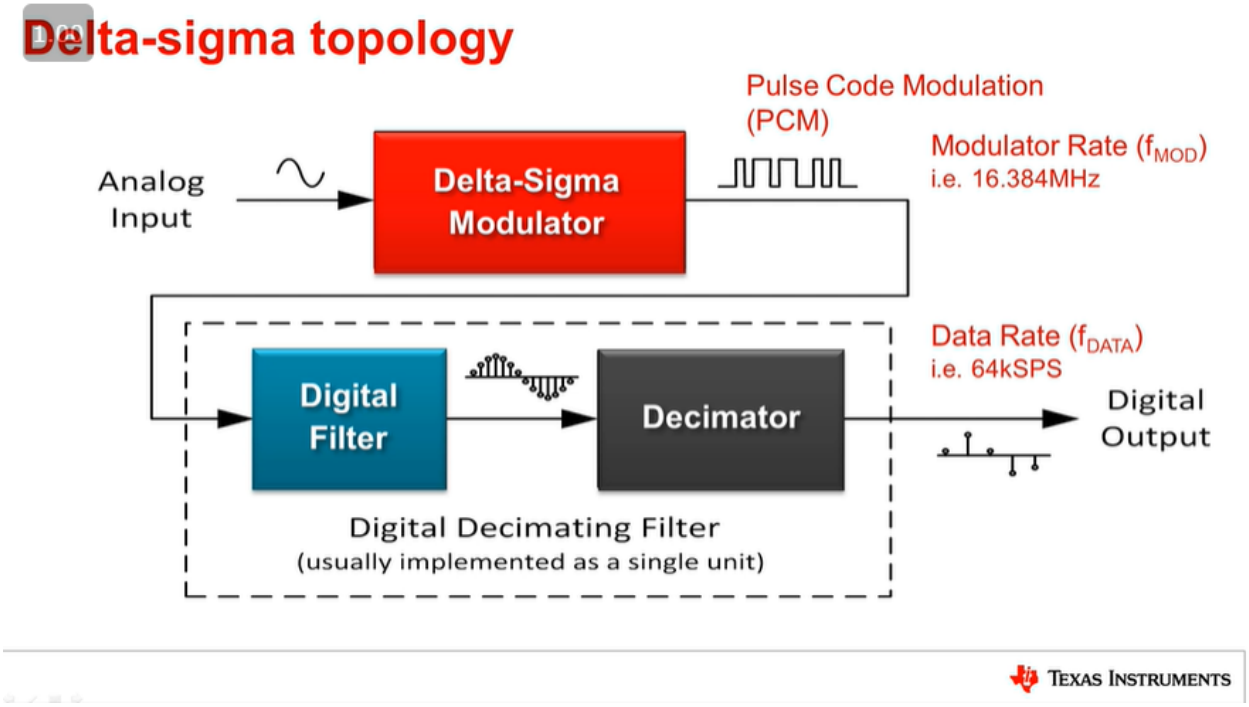
$$\frac{V_{S1} - V_{DAC}}{Ra} = \frac{V_{S1} - Vout}{Ra}$$

$$Vout = V_{DAC} = \frac{digital\ code * \left(V_{reff+} - V_{reff-}\right)}{4096}$$

Figure 1. DAC equivalent circuit



تحقیق ۱:



ADC های Delta-sigma از لحاظ نوع نمونه گیری با سایر گونه های ADC متفاوت هستند. به عنوان مثال یک SAR ADC از یک ولتاژ ورودی عکس فوری گرفته و آن را برای تعیین کد دیجیتال مربوطه تجزیه و تحلیل می کند. هر تبدیل سطح سیگنال را در آن زمان مشخص می کند. از طرف دیگر ، یک ADC دلتا سیگما سیگنال ورودی را به طور مداوم نمونه برداری می کند. پس از یک دوره زمانی خاص ، ADC یک کد دیجیتال متناسب با میانگین آن زمان صادر می کند.

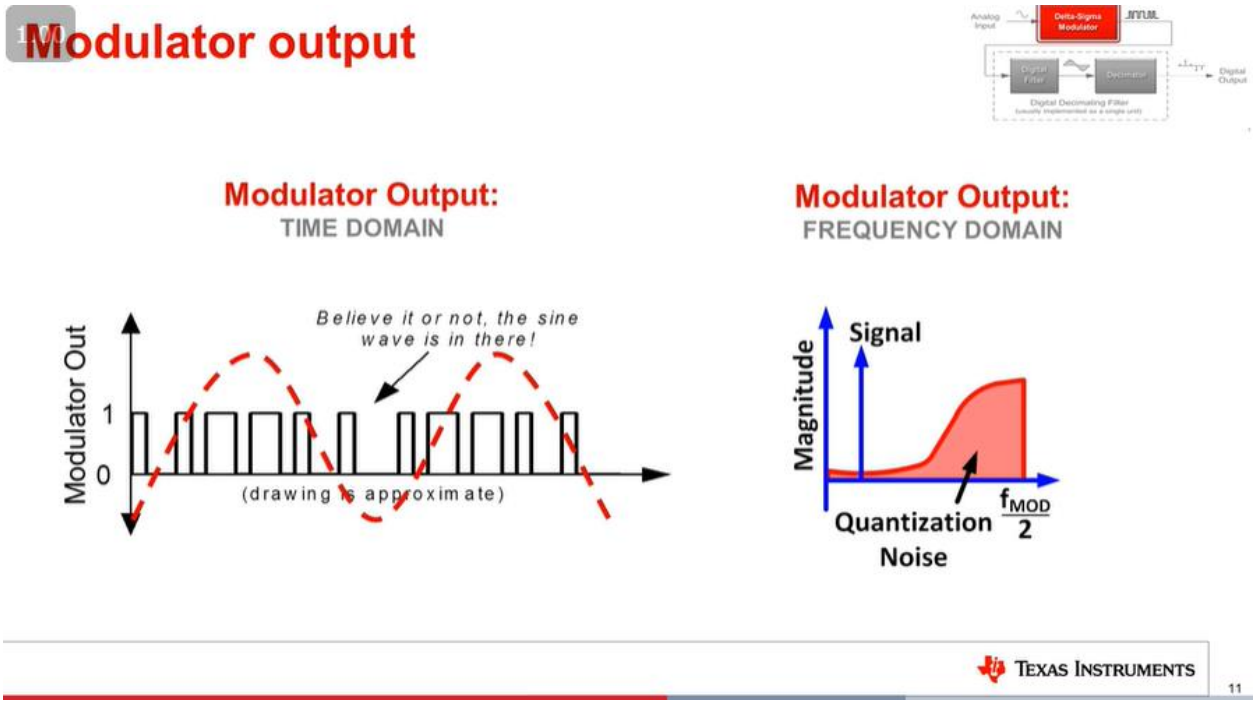
بخشی از ADC delta-sigma که ورودی آنالوگ را نمونه برداری می کند تعدیل کننده نامیده می شود. مدولاتور دلتا-سیگما شامل یک کوآنتایزر و یکپارچه کننده است که در شکل گیری صدای کمی سازی تأثیر دارد. کوآنتایزر در مدولاتور جریان ۱ و ۰ را تولید می کند که متناسب با ولتاژ ورودی آنالوگ است.

تعداد زیادی از ۱ مربوط به ولتاژ ورودی بزرگ است. این به عنوان چگالی ۱ نیز شناخته می شود. شما می توانید این را به عنوان یک تعدیل کد پالس از ورودی آنالوگ تصور کنید.

جریان bit تعدیل کننده با سرعت معروف به نرخ تعدیل کننده یا fMOD تولید می شود. به دنبال تعدیل کننده ، یک فیلتر decimation دیجیتال وجود دارد. این بخش از جریان دهنده تعدیل می کند و به نتیجه با وضوح بالاتر تبدیل می شود.

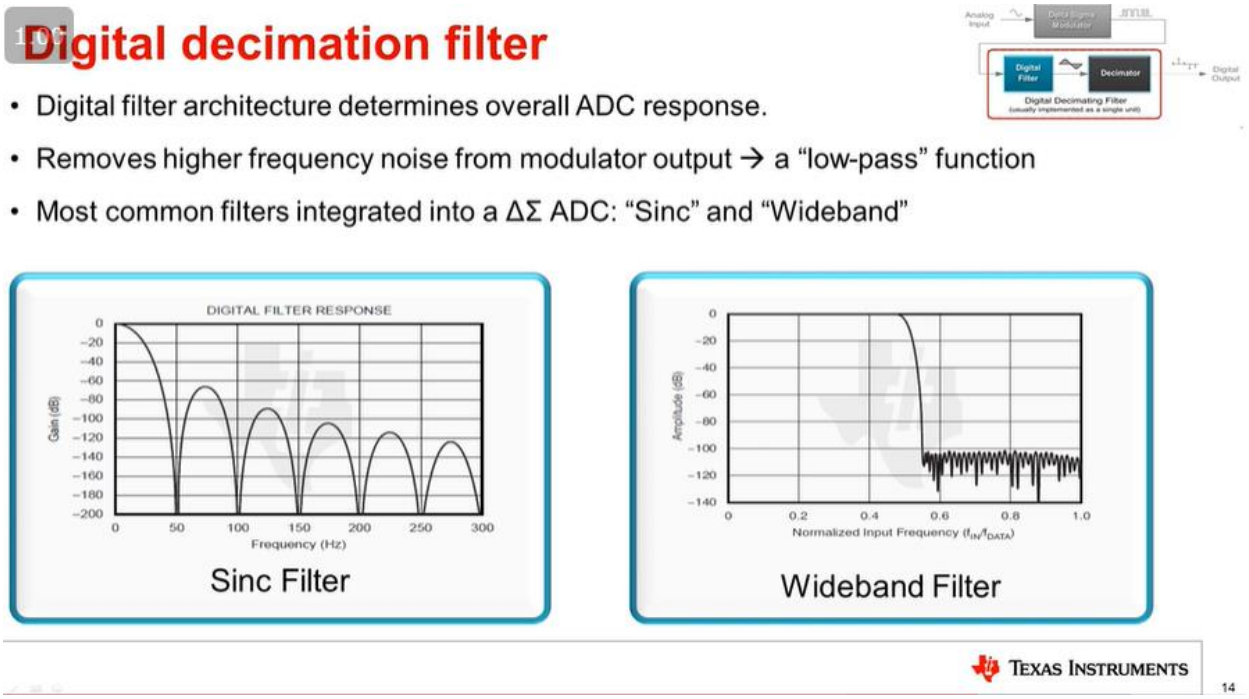
نتیجه نهایی با نرخ داده خروجی fDATA ارائه می شود. در حالی که اکثر مبدل ها فقط یک نرخ نمونه برداری دارند ، لازم به یادآوری است که ADC های delta-sigma در واقع دارای دو مورد هستند - نرخ نمونه برداری ورودی fMOD و نرخ داده خروجی .fDATAنسبت بین fMOD و fDATA را OverSampling Ratio یا OSR می نامند.

ابتدا ، بیایید نگاهی دقیق به تعدیل کننده در خروجی جریان بیت دیجیتال بیندازیم. در اینجا شکل سیگنال در خروجی تعدیل کننده در حوزه زمان و فرکانس به نظر می رسد. سیگنال دامنه زمان نسبتاً ناپایدار به نظر می رسد. با این حال ، اگر این سیگنال به طور متوسط باشد ، با مقدار ورودی اصلی آنالوگ برابر است.



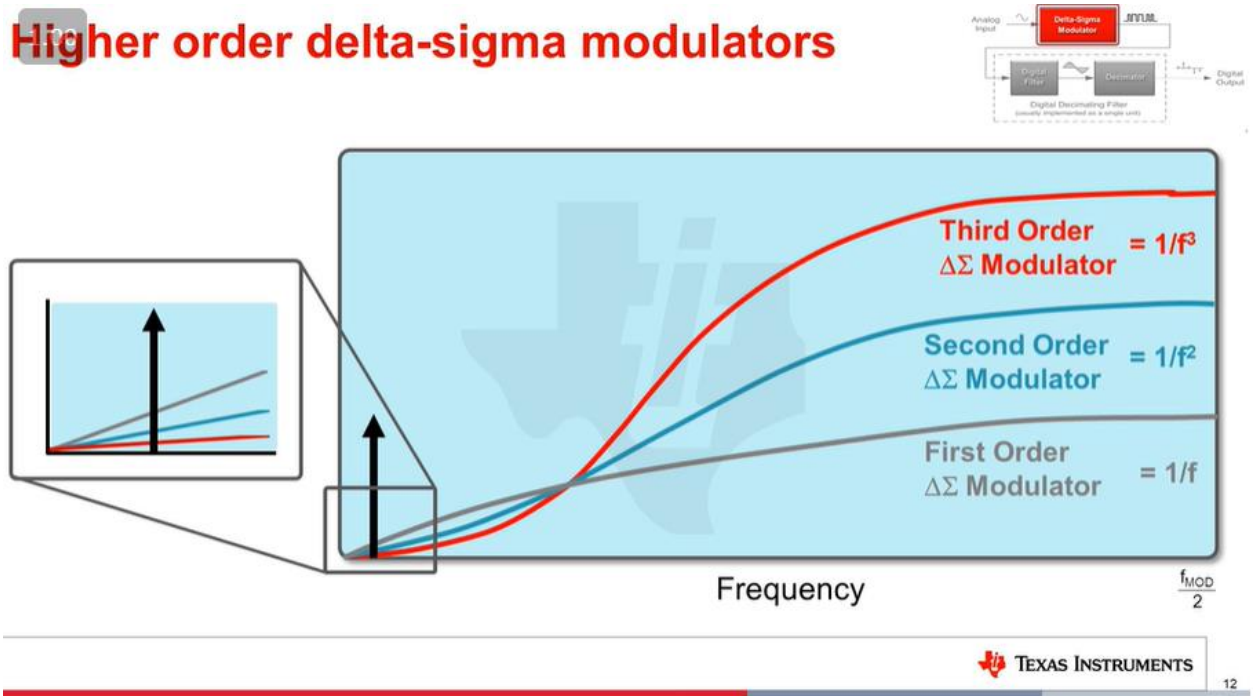
نمایش دامنه فرکانس داستان دیگری را نشان می دهد. در اینجا ، ما می توانیم قدرت سیگنال ورودی و همچنین نویز کمی را ببینیم که توسط انتگرال شکل گرفته است. اصطلاح نویز کلید عملکرد تعدیل کننده است. نویز کمی سازی برای یک مدولاتور درجه یک از صفر شروع می شود و به سرعت افزایش می یابد ، سپس در حداکثر مقدار در ۲/۱ فرکانس مدولاتور پایین می آید.

نویز یک مشکل است. و ما مطمئناً آن را در خروجی نهایی مبدل نمی خواهیم. برای خلاص شدن از شر نویز فرکانس بالاتر ، به فیلتر دیجیتال کم عبور نیاز داریم.



این نمودار تراکم مرطوب طیفی را برای یک مدولاتور مرتبه اول ، دوم و سوم با فرکانس نمونه برداری از fMOD نشان می دهد. خط قرمز در بالا پاسخ تعدیل کننده مرتبه سوم را نشان می دهد. در فرکانس های پایین نزدیک به سیگنال مورد علاقه ما ، تعدیل کننده مرتبه سوم نویز بسیار کمی است. در فرکانس های بالاتر به تدریج پر نویز می شود.

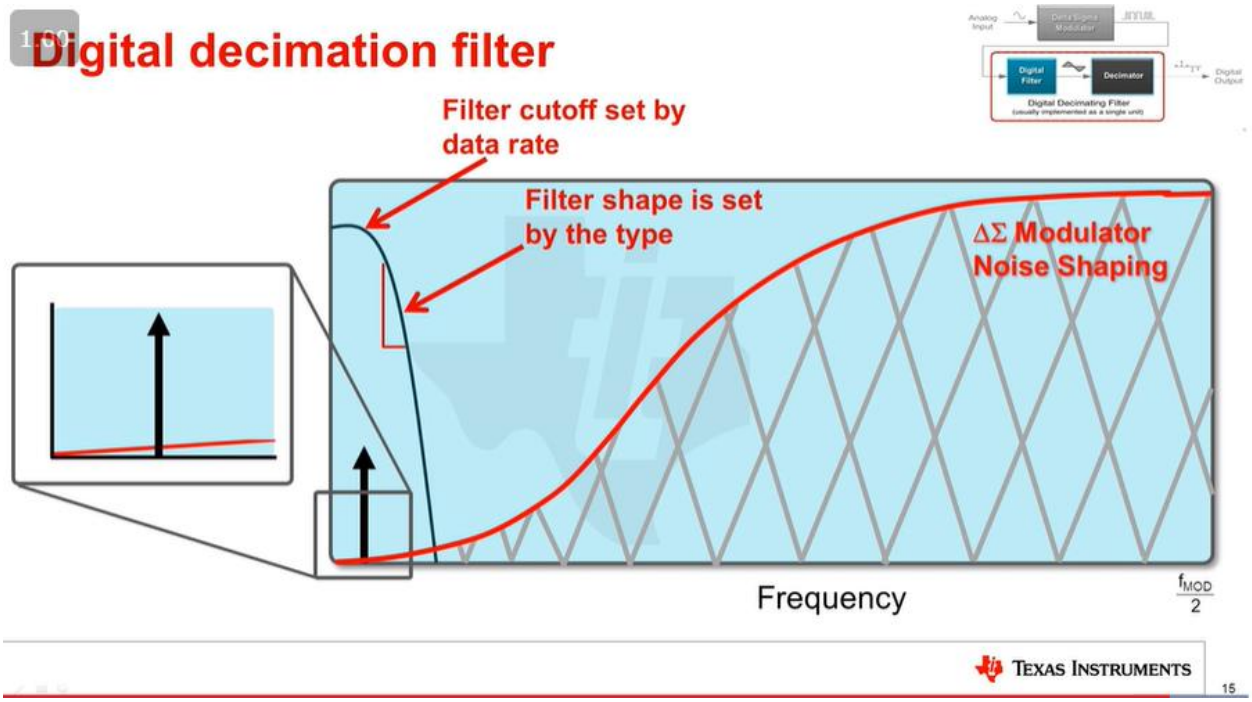
در تصویر گسترش یافته ، توجه داشته باشید که نویز کمی سازی برای یک مدولاتور درجه سوم کمتر از مدولاتور مرتبه اول شروع می شود. برای بدست آوردن نسبت های سیگنال به نویز یا SNR با نرخ داده کمتر می توان از تعدیل کننده های مرتبه بالاتر استفاده کرد.



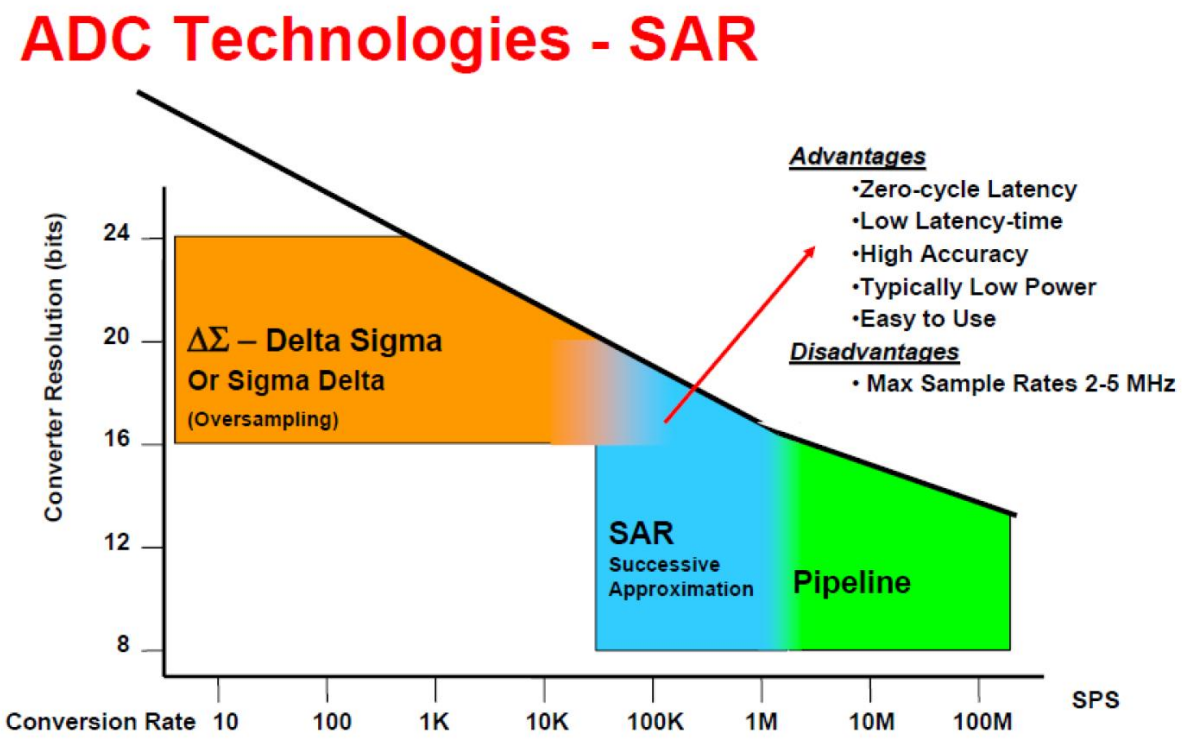
حالا ، بیایید به فیلتر decimation دیجیتال نگاه کنیم. بخاطر داشته باشید که این مرحله جریان دهنده تعدیل کننده را فیلتر و کوچک می کند. انواع مختلف فیلترها را می توان برای کاربردهای مختلف پیاده سازی کرد.

معماری فیلتر دیجیتال همان چیزی است که پاسخ کلی ADC دلتا-سیگما را تعیین می کند. پاسخ کم عبور ، نویز فرکانس بالاتر را از خروجی مدولاتور حذف می کند. در زیر دو فیلتر رایج تر وجود دارد که در ADC های دلتا-سیگما ادغام شده اند - یک فیلتر همگام سازی و فیلتر باند پهن.

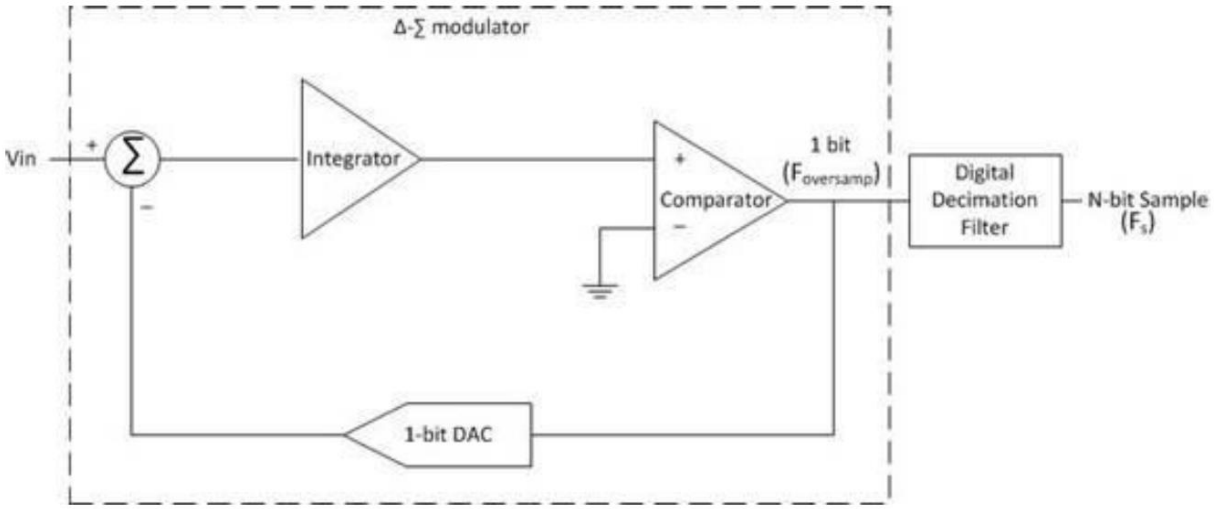
فیلتر همگام سازی در سمت چپ معمولاً در برنامه های اندازه گیری حسگر استفاده می شود. پاسخ تاخیر کم این فیلتر به ورودی های یک ADC اجازه می دهد تا با تنظیم سریع بین چندین سنسور جابجا شوند. فیلتر باند پهن در سمت راست بیشتر در برنامه های AC دیده می شود که به تجزیه و تحلیل دامنه فرکانس مداوم نیاز دارند.



همانطور که گفته شد ، برای از بین بردن بیشتر نویز کوانتاسیون تعدیل کننده ، به پاسخ کم عبور فیلتر دیجیتال نیاز است. شکل فیلتر توسط نوع فیلتر تنظیم می شود. در حالی که فرکانس قطع فیلتر با نرخ داده خروجی تنظیم می شود. برای یک فیلتر sync3 ، فرکانس قطع تقریباً ۴/۱ نرخ داده است. برای یک فیلتر باند پهن ، فرکانس قطع تقریباً ۲/۱ نرخ داده است.



این نوع مبدل ها برای کاربرد های با رزولویشن بالا و نرخ نمونه برداری تا چند صد هرتز مورد استفاده قرار میگیرند . از این نوع مبدل ها در سنسور های میدان مغناطیسی ، سنسور های دما و ... استفاده میشود و کاربردهای مختلفی ئر نمونه برداری های صوتی دارند .مبدل دلتاسیگما تاخیر بزرگتری نسبت به SAR ADC در تحویل مقدار تبدیل شده دارد چون این مبدل چند بار از ورودی در یک محدوده زمانی از پیش تعیین شده به سرعت نمونه برداری میکند و سپس مقدار میانگین بدست آمده را تحویل خروجی میدهد .این نوع ADC به علت نرخ کلاک بالا مصرف ارژی بیشتری نسبت به SAR ADC دارد . تکنیک میان گیری در این مبدل یک روش دیجیتالی برای فیلتر کردن می‌باشد در نتیجه مبدل فقط نیاز به یک فیلتر پایین گذر خارجی برای ورودی های آنالوگ میباشد .که این فیلتر بخوبی در مقابل نویز مقاومت می کند[۱].



STM32f4xx از مدل تقریب متوالی (SAR) استفاده میکند .

تحقیق ۲:

درمورد مفاهیم مربوط به نرخ نمونه برداری، زمان تبدیل، رزولوشن، خطای کوانتیزه کردن و ولتاژ مرجع در مبدل های آنالوگ به دیجیتال تحقیق و گزارش نمایید.

یک سیگنال آنالوگ پیوسته در زمان و لازم است که این مورد به جریان مقادیر دیجیتال تبدیل شود. بنابراین لازم است که سرعت نمونه برداری از مقادیر دیجیتالی جدید از سیگنال آنالوگ تعریف شود. به نرخ مقادیر جدید میزان نمونه برداری یا فرکانس نمونه برداری مبدل گفته می شود. می توان یک سیگنال باند محدود به طور مداوم متغیر ۲۹۸> نمونه برداری کرد و سپس سیگنال اصلی را می توان از مقادیر زمان گسسته توسط فیلتر بازسازی تولید کرد. قضیه نمونه برداری Nyquist – Shannon نشان می دهد که تولید مثل صادقانه سیگنال اصلی تنها در صورتی امکان پذیر است که نرخ نمونه برداری از دو برابر بالاترین فرکانس سیگنال بالاتر باشد.

از آنجا که یک ADC عملی نمی تواند یک لحظه تبدیل کند ، مقدار ورودی باید لزوماً در مدتی که مبدل یک تبدیل را انجام می دهد ثابت باشد (به نام زمان تبدیل). مدار ورودی به نام نمونه و نگه داشتن این وظیفه را انجام می دهد - در بیشتر موارد با استفاده از خازن برای ذخیره ولتاژ آنالوگ در ورودی و استفاده از سوئیچ یا گیت الکترونیکی برای

قطع اتصال خازن از ورودی. بسیاری از مدارهای مجتمع ADC شامل این نمونه هستند و سیستم فرعی را به صورت داخلی نگه می دارند.

زمان تبدیل به زمانی گفته می‌شود که ADC در هر روشی از زمان شروع تبدیل تا نتیجه نهایی زمان صرف می‌کند.

سرعت ADC براساس نوع متفاوت است. Wilkinson ADC با سرعت کلاک محدود می شود که توسط مدارهای دیجیتال فعلی قابل پردازش است. برای تقریب متوالی ADC ، زمان تبدیل با لگاریتم رزولوشن ، یعنی تعداد بیت ها ، مقیاس می شود. Flash ADCs مطمئناً سریعترین نوع از این سه مورد است. تبدیل اساساً در یک مرحله موازی انجام می شود.

بین سرعت و دقت احتمال بالقوه وجود دارد. ADC های فلش دارای رانش و عدم قطعیت در ارتباط با نتایج مقایسه کننده در خطی ضعیف هستند. به میزان کمتری ، خطی بودن ضعیف نیز می تواند مسئله ADC های تقریب پی در پی باشد. در اینجا ، غیرخطی بودن از تجمع خطاهای حاصل از فرایندهای تفریق ناشی می شود. ADC های ویلکینسون دارای بهترین خطی بودن از این سه مورد هستند

خطای کوانتیزه سازی توسط مقداردهی ذاتی یک ADC ایده آل است. این یک خطای گردی است بین ولتاژ ورودی آنالوگ به ADC و مقدار دیجیتالی خروجی. خطا غیر خطی و وابسته به سیگنال است. در یک ADC ایده آل ، جایی که خطای کوانتیزاسیون به طور یکنواخت بین −۱/۲ LSB و +۱/۲ LSB توزیع می شود ، و سیگنال توزیع یکنواختی دارد که تمام سطوح کوانتیزاسیون را پوشش می دهد ، نسبت سیگنال به کوانتیزاسیون و نویز (SQNR) توسط

$$\mathrm{SQNR} = 20\log_{10}(2^Q) \approx 6.02 \cdot Q \text{ dB}$$

که در آن Q تعداد بیت های کوانتیزاسیون است. به عنوان مثال ، برای 16 ADC بیتی ، خطای کمیت ۹۶.۳ دسی بل زیر حداکثر سطح است.

خطای اندازه گیری از DC به فرکانس Nyquist توزیع می شود. در نتیجه ، اگر از بخشی از پهنای باند ADC استفاده نشود ، مانند مورد نمونه برداری بیش از ۱۲۸>، برخی از خطاهای اندازه گیری خارج از باند رخ می دهد ، به طور موثر SQNR را برای پهنای باند در حال استفاده در یک سیستم بدون نمونه ، می توان از نویز دهی برای افزایش بیشتر SQNR با مجبور کردن خطای کوانتاسیون بیشتر از باند استفاده کرد.Wikipedia site:fa.tr2tr.wiki

Vrefیا همان ولتاژ مرجع درADC :

این ولتاژ برای مقایسه با ولتاژ ورودی استفاده میشود. مثلا اگر ولتاژ مرجع را ۴ انتخاب کنیم ، بازه ۰-۴ نیز به ۱۰۲۴ قسمت تقسیم میشود.

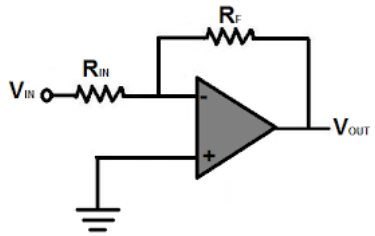
تحقیق ۳:

وقتی یک پین به عنوان یک ورودی مبدل آنالوگ به دیجیتال انتخاب می‌شود، این پین به مدار Sample and hold وصل می‌شود. وظیفه این مدار نمونه برداری از ولتاژ لحظه ای ورودی و ثابت نگه داشتن آن تا نمونه برداری بعدی است. درصورتی که نیاز به خواندن ولتاژ از چندین کانال ورودی به وطور همزمان باشد، این مدار این امکان را به کاربر می دهد تا نمونه‌برداری بعدی، ولتاژ تمام کانالها را ثابت نگه دارد. اگر مدار S/H وجود نداشت، برای خواندن همزمان تمام کانالها به همان تعداد ADC نیاز بود.

اگر یک منبع سیگنال آنالوگ با امپدانس خروجی بالا به کارگرفته شود، ثابت زمانی)ضرب امپدانس در ظرفیت خازن(را افزایش میدهد. وقتی ثابت زمانی افزایش یابد، شارژشدن خازن به مدت زمان بیشتری نیاز خواهد داشت و با توجه به اینکه ولتاژ خازن درواقع همان نمونه ای است که از سیگنال آنالوگ ورودی گرفته میشود در کار نمونه برداری اخلال ایجاد شده و نمونه برداری به درستی انجام نخواهد شد.

تحقیق ۴:

می‌توان با استفاده از یک وارون ساز ساخته شده با اپ امپ ولتاژ را قرینه کرد.



تحقیق ۴:

. مثلاً فرض کنید که ولتاژ رفرنس و تغذیه رو از رگولاتوری می‌گیریم که یه بار ۳.۳ میده، یک بار ۳.۱ میده و .... حال در این حالت هر بار ولتاژ تغذیه رو با ADC بخوانیم، ۴۰۹۵ می‌گیریم در حالی که ولتاژ تغییر کرده.

یک راه حل برای این موضوع این هست که ولتاژ مرجع را روی رفرنس داخلی بگذاریم و ولتاژ تغذیه را با ADC بخوانیم. بعد مجدد رفرنس را به Vref برگردونید. با این تفاوت که در محاسباتمان الآن دقیقاً می‌دانیم که Vref چقدر هست.

آزمایش ۱:

```
#include "main.h"
#include "stm32f4xx_hal.h"
#include "stm32f429i_discovery_lcd.h"
```

```
#include "stdio.h"


int main(void)
{
	int v;
	float angle, vout,vref = 3.3;
	char vstr[6],anglestr[6];

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_CRC_Init();
  MX_DMA2D_Init();
  MX_FMC_Init();
```

```
 MX_I2C3_Init();
 MX_LTDC_Init();
 MX_SPI5_Init();
 MX_TIM1_Init();
 MX_USART1_UART_Init();
 MX_ADC1_Init();
 /* USER CODE BEGIN 2 */


 /* USER CODE END 2 */


 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
        BSP_LCD_Init();
 BSP_LCD_LayerDefaultInit(LCD_BACKGROUND_LAYER, LCD_FRAME_BUFFER);
 BSP_LCD_LayerDefaultInit(LCD_FOREGROUND_LAYER, LCD_FRAME_BUFFER);
 BSP_LCD_SelectLayer(LCD_FOREGROUND_LAYER);
 BSP_LCD_DisplayOn();


        BSP_LCD_Clear(0xFFFFFFFF);



while (1)
   {
                        HAL_ADC_Start( &hadc1);
                        HAL_ADC_PollForConversion(&hadc1,200);
                        v = HAL_ADC_GetValue(&hadc1);
                        vout =   (v * vref)/4095;
                        angle = vout * 270        /3.3;
                        sprintf(vstr,"%3.3f",vout);
                        sprintf(anglestr,"%3.3f",angle);
                        BSP_LCD_DisplayStringAt(20,20,vstr,CENTER_MODE);
```

```
                BSP_LCD_DisplayStringAt(20,20,"     ",CENTER_MODE);
                BSP_LCD_DisplayStringAt(20,20,anglestr,CENTER_MODE);
            }
}
```

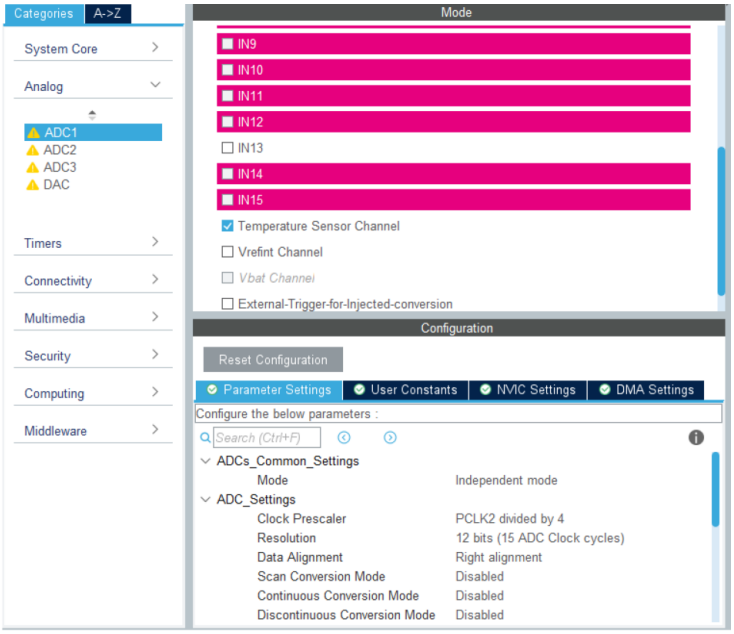
```
while (1)
    {
      HAL_ADC_Start( &hadc1);
      HAL_ADC_PollForConversion(&hadc1,200);
      v = HAL_ADC_GetValue(&hadc1);
      vout =  (v * vref)/4095;
      angle = vout * 270   /3.3;
      sprintf(vstr,"%3.3f",vout);
      sprintf(anglestr,"%3.3f",angle);
      BSP_LCD_DisplayStringAt(20,20,vstr,CENTER_MODE);
      BSP_LCD_DisplayStringAt(20,20,"     ",CENTER_MODE);
      BSP_LCD_DisplayStringAt(20,20,anglestr,CENTER_MODE);
    }
```

آزمایش ۲:

```
  BSP_LCD_Clear(0xFFFFFFFF);


while (1)
    {
      BSP_LCD_Clear(0xFFFFFFFF);
      BSP_LCD_DrawHLine(20,300,200);
      BSP_LCD_DrawHLine(20,100,200);

      HAL_ADC_Start( &hadc1);
      HAL_ADC_PollForConversion(&hadc1,200);
      v = HAL_ADC_GetValue(&hadc1);
      vout =  (v * vref)/4095;
      prevtemp = (vout/0.01)+2;
      sprintf(tempstr,"%3.3f",prevtemp);
      BSP_LCD_DisplayStringAt(20,20,tempstr,CENTER_MODE);

      for ( i=0;i<20;i++){
        HAL_ADC_Start( &hadc1);
        HAL_ADC_PollForConversion(&hadc1,200);
        v = HAL_ADC_GetValue(&hadc1);
        vout =  (v * vref)/4095;
        prevtemp = (vout/0.01)+2;
        sprintf(tempstr,"%3.3f",prevtemp);
        BSP_LCD_DisplayStringAt(20,20,tempstr,CENTER_MODE);
        BSP_LCD_DrawLine(10*i+10,300-(20*temp),10*(i+1)+20,300-(20*prevtemp));
        prevtemp = temp;

      }
```

آزمایش ۳:

کد این قسمت مشابه قسمت قبل است فقط در قسمت تنظیمات به جای این که ورودی از IN5 باشد، از Temperature Sensor Channel خواهد بود.



آزمایش ۴:

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DAC_Init();
  /* USER CODE BEGIN 2 */
  int i;
  HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
  HAL_Delay(1);
  const unsigned short ECG[94] = {
0x3E0,  0x3E8,  0x460,  0x490,  0x498,  0x538,  0x548,  0x4D0,  0x478,  0x418,
0x408,  0x338,  0x330,  0x300,  0x340,  0x310,  0x338,  0x5C0,  0xB90,  0xF20,
0x558,  0x0,  0x40,  0x1D8,  0x370,  0x360,  0x388,  0x378,  0x3A0,  0x398,
0x3D0,  0x3D0,  0x408,  0x420,  0x448,  0x460,  0x490,  0x4B0,  0x4F8,  0x510,
0x568,  0x590,  0x5D0,  0x5E8,  0x5F8,  0x5F8,  0x608,  0x5F8,  0x5D0,  0x588,
0x548,  0x500,  0x4C0,  0x488,  0x488,  0x470,  0x470,  0x470,  0x478,  0x470,
0x488,  0x480,  0x480,  0x480,  0x478,  0x460,  0x460,  0x450,  0x448,  0x440,
0x430,  0x428,  0x428,  0x418,  0x420,  0x418,  0x418,  0x408,  0x418,  0x408,
0x418,  0x400,  0x410,  0x400,  0x408,  0x3E0,  0x3F8,  0x3D0,  0x3E8,  0x3D0,
0x3F0,  0x3E8,  0x3F8,  0x3E0, };


  while (1)
  {
    for(i = 0;i<94;i++){
        HAL_DAC_SetValue(&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,ECG[i]); // set DAC value channel 1 as i th component of ECG
        HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
        HAL_Delay(1);
    }
  }
}
```

آزمایش ۵:

```
  while (1)
  {
    for(i = 0;i<17450;i++){
        HAL_DAC_SetValue(&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,array[i]); // set DAC value channel 1 as i th component of voice
        HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
        HAL_Delay(1);
    }
  }
}
```

مراجع

[1] https://www.ti.com/

https://fa.tr2tr.wiki/wiki/Analog-to-digital_converter