MICROS 32 BITS
STM - DAC

ROBINSON JIMENEZ MORENO

LUISA FERNANDA GARCIA VARGAS

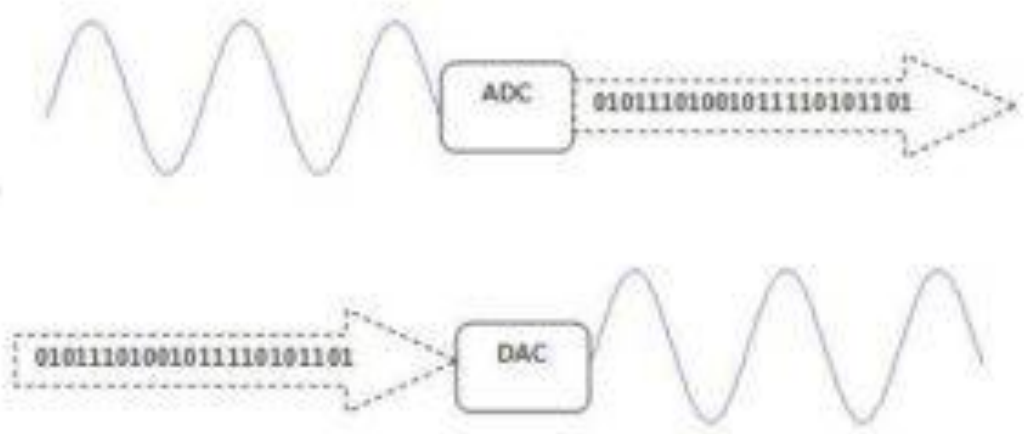
UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

DAC - STM32F4

Introducción:

DAC (Digital-to-Analogue Converter) es el módulo encargado de convertir valores binarios digitales en salidas de voltaje análogo. Entre los usos de este módulo se encuentra la generación de señales de audio y la generación de formas de onda, entre otros.



https://www.culturasonora.es/hifi/que-es-un-dac/
http://embedded-lab.com/blog/stm32-digital-analogue-converter-dac/

STM32F4x - datasheet

DAC - STM32F4

Introducción:

El módulo DAC de la STM32F4x se puede configurar en modo de 8 o 12 bits. En el modo de 12 bits, los datos pueden alinearse a la izquierda o a la derecha. El DAC tiene dos canales de salida, cada uno con su propio convertidor. En el modo de canal dual DAC, las conversiones se pueden realizar de forma independiente o simultánea.

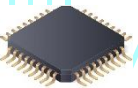
http://embedded-lab.com/blog/stm32-digital-analogue-converter-dac/

STM32F4x - datasheet

DAC - STM32F4

Características generales:

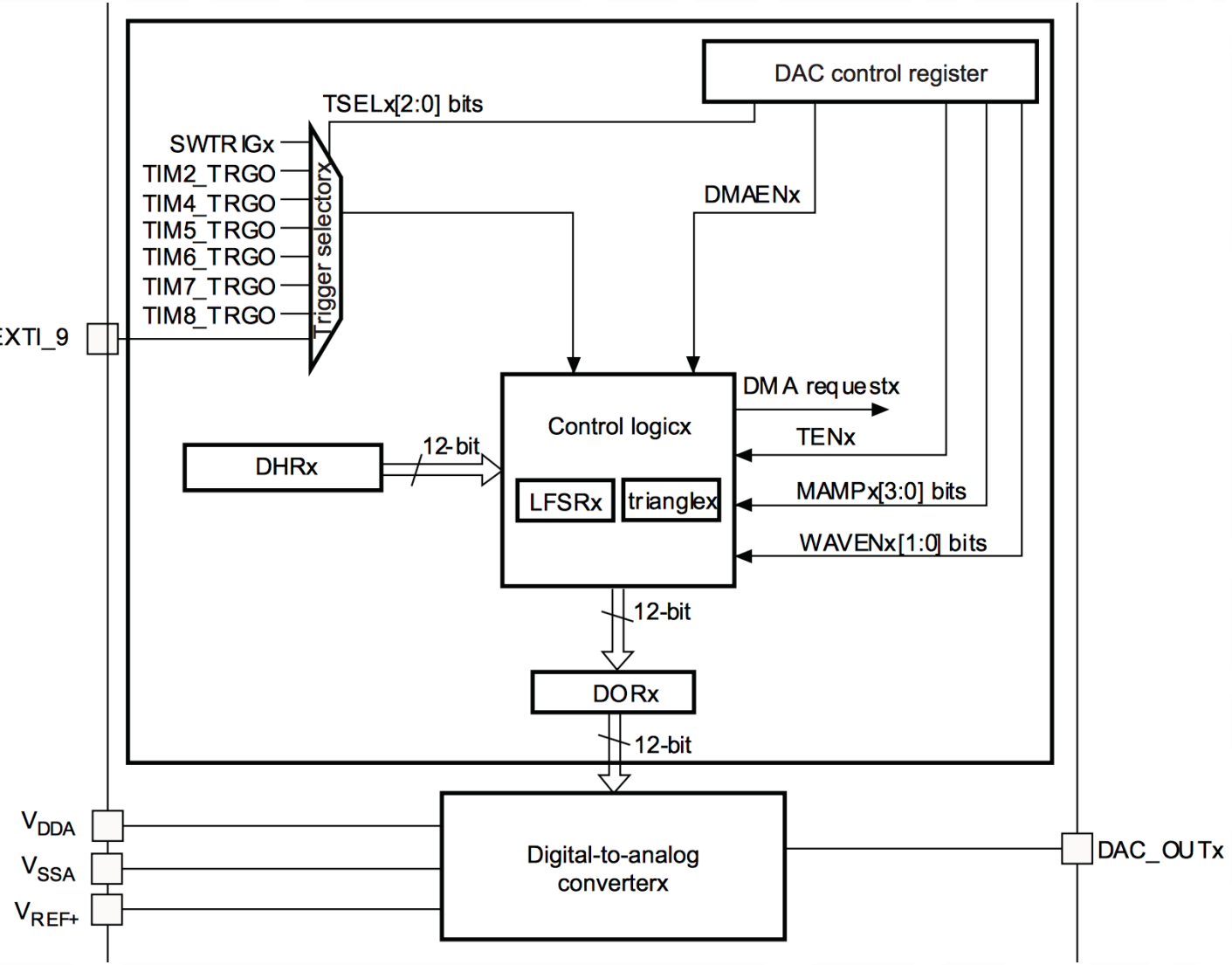
- Dos convertidores DAC: un canal de salida para cada uno
- Alineación de datos izquierda o derecha en modo de 12 bits
- Capacidad de actualización sincronizada
- Generación de ondas de ruido
- Generación de onda triangular
- Canal Dual DAC para conversiones independientes o simultáneas
- Referencias de voltaje externos para la conversión
- Referencia de voltaje de entrada VREF +



STM32F4x - datasheet

Introducción - Módulo DAC de la STM32F4x:

Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).



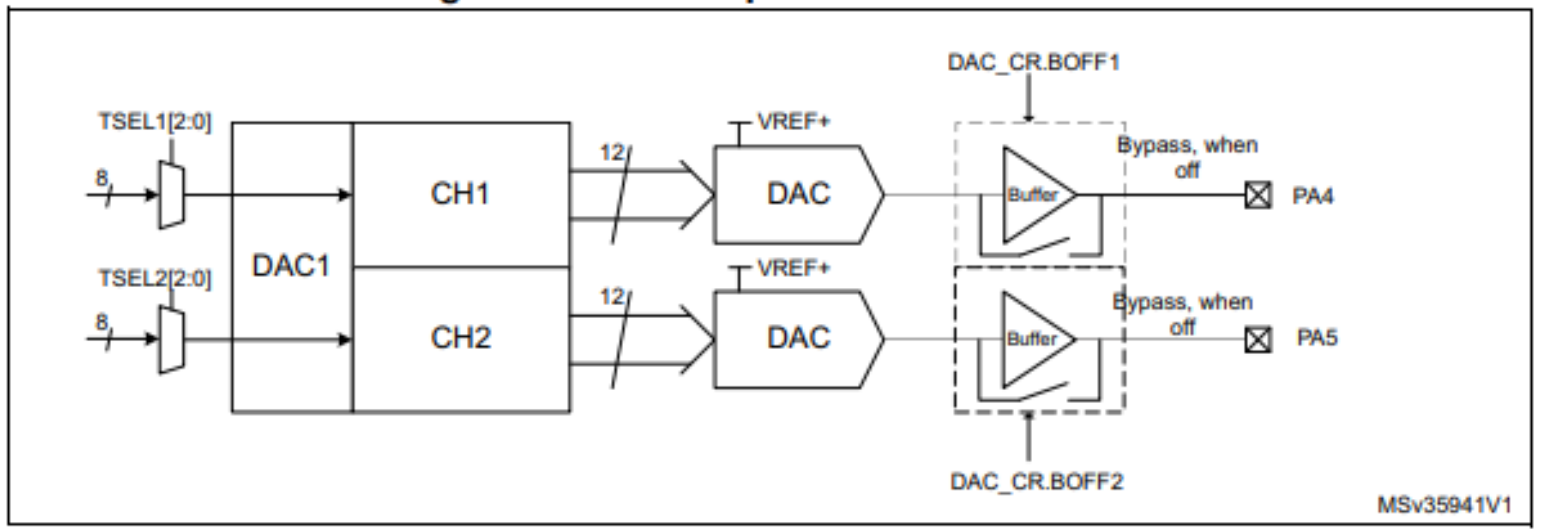
STM32F4x - datasheet

Introducción - Módulo DAC de la STM32F4x:

16.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

Figure 94. DAC output buffer connection



Introducción – DAC, pines:

Una vez que el canal del DAC está habilitado, el pin GPIO correspondiente (PA4 o PA5) se conecta automáticamente a la salida del conversor analógico (DAC_OUTx). El pin de referencia de entrada, VREF+ (compartido con ADC) está disponible para una mejor resolución.

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, 1.8 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

STM32F4x - datasheet

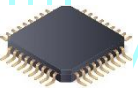
DAC - STM32F4

Funcionamiento:

1-Habilitar canal DAC: Cada canal DAC puede encenderse configurando su correspondiente bit ENx en el registro DAC_CR.

2-Habilitar buffer de salida DAC: Cada búfer de salida del canal DAC puede habilitarse y deshabilitarse utilizando el bit BOFFx correspondiente en el registro DAC_CR.

3-Formato de datos DAC: Dependiendo del modo de configuración seleccionado (8 o 12 bits), los datos deben escribirse en el registro especificado como se indica a continuación:



STM32F4x - datasheet

DAC - STM32F4

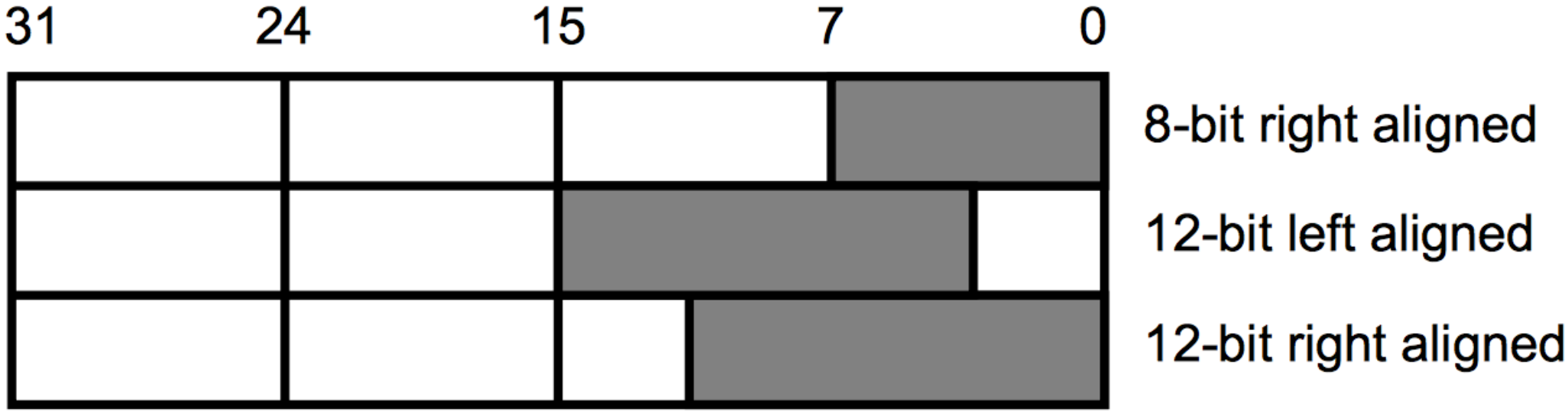
Funcionamiento:

Formato de datos DAC – Single DAC Channel:

- Alineación derecha de **8 bits**: el software tiene que cargar datos en el DAC_DHR8Rx [7: 0]bits (almacenados en los bits DHRx [11: 4])
- Alineación **izquierda de 12 bits**: el software tiene que cargar datos en los bits DAC_DHR12Lx [15: 4] (almacenados en los bits DHRx [11: 0])
- Alineación **derecha de 12 bits**: el software tiene que cargar datos en los bits DAC_DHR12Rx [11: 0] (almacenados en los bits DHRx [11: 0])

Dependiendo del registro DAC_DHRx, los datos escritos por el usuario se desplazan y se almacenan en el DHRx correspondiente. El registro DHRx luego se carga en el registro **DORx.**

DAC - STM32F4



STM32F4x - datasheet

DAC - STM32F4

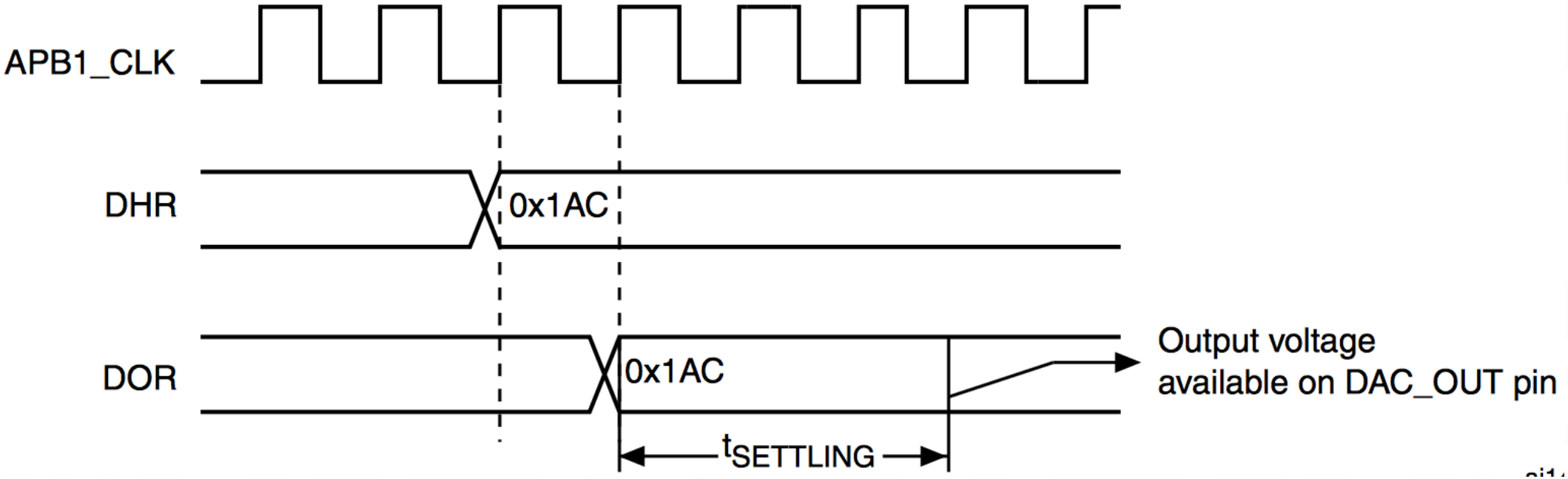
Funcionamiento:

Conversión DAC: cualquier transferencia de datos al canalx del DAC se debe realizar cargando el registro DAC_DHRx. Los datos almacenados en el registro DAC_DHRx se cargan automáticamente en el registro DAC_DORx. (DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx)

Cuando DAC_DORx se carga con el contenido de DAC_DHRx, el voltaje análogo de salida queda disponible después de un tiempo, que depende del voltaje de la fuente de alimentación y de la carga de salida analógica, en función al clock del puerto ajustado en el registro APB.

DAC - STM32F4

Funcionamiento – *Conversión DAC:*



STM32F4x - datasheet

DAC - STM32F4

Voltaje de salida DAC: Los valores digitales se convierten a voltajes de salida en una conversión lineal entre 0 y VREF .

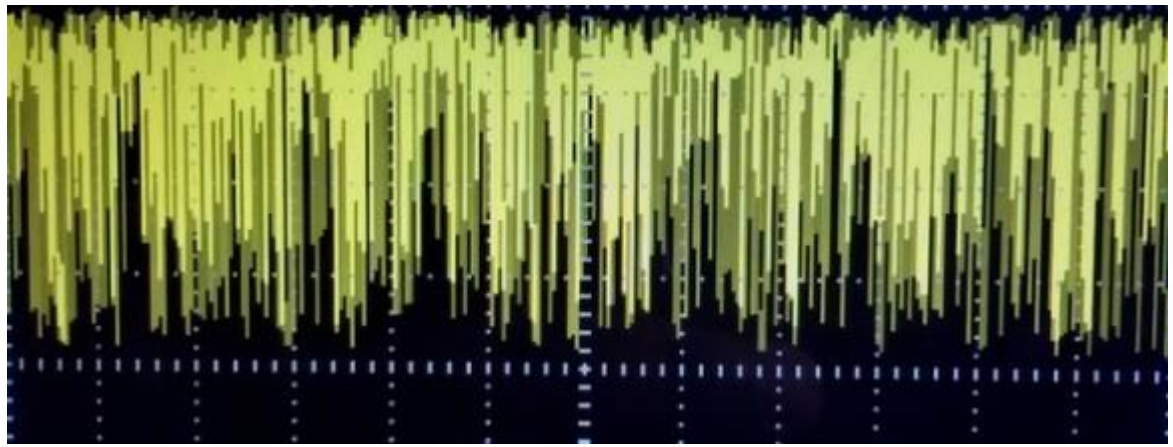
$$V_{out} = \frac{Digital\,Value\,(DAC\_DHRx) \times V_{Ref}}{(2^n - 1)}$$

$$\text{DACoutput} = V_{REF} \times \frac{\text{DOR}}{4096}$$

STM32F4x - datasheet

GENERACIÓN DE RUIDO

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to "01". The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.



The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

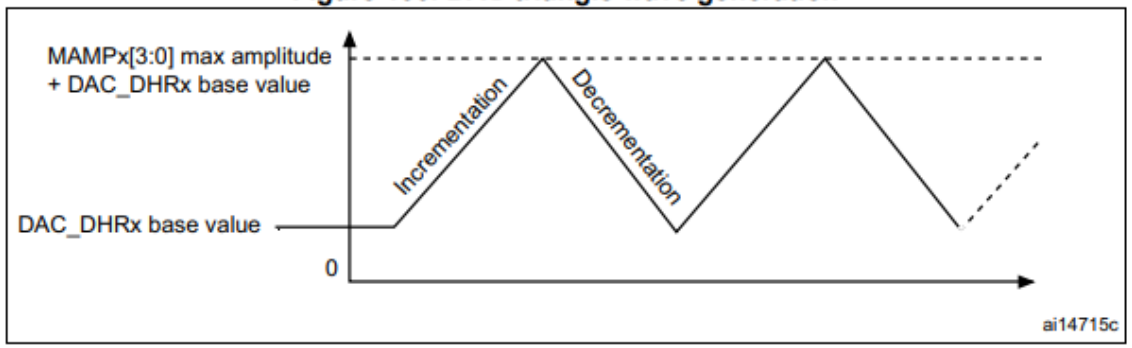
http://embedded-lab.com/blog/stm32-digital-analogue-converter-dac/

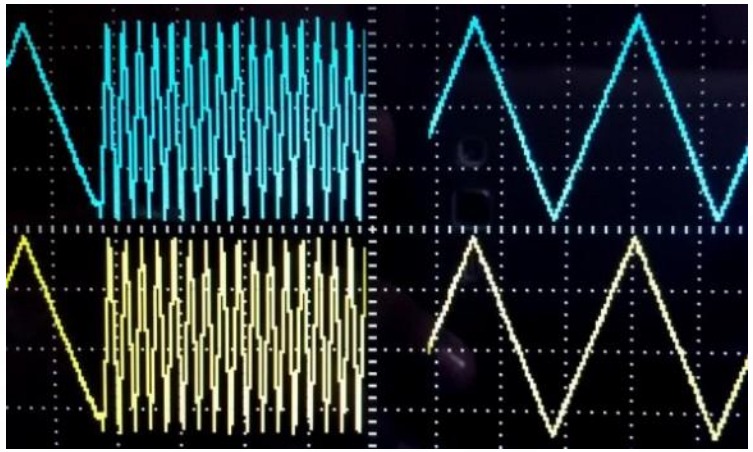
GENERACIÓN ONDA TRIÁNGULO

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register.

The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register. The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

Figure 100. DAC triangle wave generation





http://embedded-lab.com/blog/stm32-digital-analogue-converter-dac/

DAC - STM32F4

DAC_CR:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm

DAC - STM32F4

DAC_CR:

Bit 29 – DMAUDRIE2: Habilitar DAC channel2 DMA underrun interrupt

0: Deshabilitado

1: Habilitado

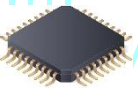
Bit 28 – DMAEN2: Habilita el DAC channel2 DMA

0: Deshabilitado

1: Habilitado

Bit 27:24 – MAMP2[3:0]: selecciona la máscara en modo de generación de onda o la amplitud en modo de generación de triángulo.

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm



DAC - STM32F4

DAC_CR:

Bit 27:24 – MAMP2[3:0]

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

DAC - STM32F

DAC_CR:

Bit 23:22 – WAVE2[1:0]: Habilitar generación de ruido u onda triangular en DAC channel2 (Solo se utiliza si el bit TEN2 = 1)

00: Deshabilitar generación de onda

01: Habilitar generación de ruido

1x: Habilitar generación de onda triangular

Bit 21:19 – TSEL2[2:0]: Estos bits seleccionan el evento externo utilizado para activar el DAC channel2 (Solo se utiliza si el bit TEN2 = 1)

000: Timer 6 TRGO event	100: Timer 2 TRGO event
001: Timer 8 TRGO event	101: Timer 4 TRGO event
010: Timer 7 TRGO event	110: External line9
011: Timer 5 TRGO event	111: Software trigger

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm

DAC - STM32F4

DAC_CR:

Bit 18 – TEN2: habilita/deshabilita el disparador (trigger) DAC channel2

0: DAC channel2 trigger deshabilitado y los datos escritos en el registro DAC_DHRx se transfieren un ciclo de reloj APB1 más tarde al registro DAC_DOR2

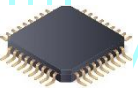
1: DAC channel2 trigger habilitado y los datos del registro DAC_DHRx se transfieren tres ciclos de reloj APB1 más tarde al registro DAC_DOR2

Bit 17 – BOFF2: habilita/deshabilita el buffer de salida DAC channel2.

0: Habilitado

1: Deshabilitado

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm



DAC - STM32F4

DAC_CR:

Bit 16 – EN2: habilita/deshabilita DAC channel2

0: Deshabilitado

1: Habilitado

Bit 15:14: Reservado

Bit 13 – DMAUDRIE1: Habilitar DAC channel1 DMA underrun interrupt

0: Deshabilitado

1: Habilitado

Bit 12 – DMAEN1: Habilita el DAC channel1 DMA

0: Deshabilitado

1: Habilitado

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm

DAC - STM32F4

DAC_CR:

Bit 11:8 – MAMP1[3:0]: selecciona la máscara en modo de generación de onda o la amplitud en modo de generación de triángulo.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

DAC - STM32F4

DAC_CR:

Bit 7:6 – WAVE1[1:0]: Habilitar generación de ruido u onda triangular en DAC channel1 (Solo se utiliza si el bit TEN1 = 1)

00: Deshabilitar generación de onda

01: Habilitar generación de ruido

1x: Habilitar generación de onda triangular

Bit 5:3 – TSEL1[2:0]: Estos bits seleccionan el evento externo utilizado para activar el DAC channel1 (Solo se utiliza si el bit TEN1 = 1)

000: Timer 6 TRGO event	100: Timer 2 TRGO event
001: Timer 8 TRGO event	101: Timer 4 TRGO event
010: Timer 7 TRGO event	110: External line9
011: Timer 5 TRGO event	111: Software trigger

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm

DAC - STM32F4

DAC_CR:

Bit 2 – TEN1: habilita/deshabilita el disparador (trigger) DAC channel1

0: DAC channel1 trigger deshabilitado y los datos escritos en el registro DAC_DHRx se transfieren un ciclo de reloj APB1 más tarde al registro DAC_DOR1

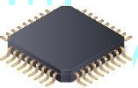
1: DAC channel1 trigger habilitado y los datos del registro DAC_DHRx se transfieren tres ciclos de reloj APB1 más tarde al registro DAC_DOR1

Bit 1 – BOFF1: habilita/deshabilita el buffer de salida DAC channel1.

0: Habilitado

1: Deshabilitado

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm



DAC - STM32F4

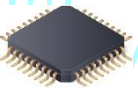
DAC_CR:

Bit 0 – EN1: habilita/deshabilita DAC channel1

0: Deshabilitado

1: Habilitado

http://download.mikroe.com/documents/compilers/mikroc/arm/help/i2c_library.htm



DAC - STM32F4

DAC_DHR12R1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 – DACC1DHR[11:0]: DAC channel1 12-bits datos, alineados a la derecha

DAC - STM32F4

DAC_DHR12L1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bit 15:4 – DACC1DHR[11:0]: DAC channel1 12 bits-datos alineados a la izquierda

Bit 3:0: Reservado

DAC - STM32F4

DAC_DHR8R1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 – DACC1DHR[7:0]: DAC channel1 8-bits datos alineados a la derecha

DAC - STM32F4

DAC_DHR12R2

•

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 – DACC2DHR[11:0]: DAC channel2 12-bits datos alineados a la derecha

DAC - STM32F4

DAC_DHR12L2:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bit 15:4 – DACC2DHR[11:0]: DAC channel2 12 bits-datos alineados a la izquierda

Bit 3:0: Reservado

DAC - STM32F4

DAC_DHR8R2:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 – DACC2DHR[7:0]: DAC channel2 8-bits datos alineados a la derecha

F4x - Datasheet

DAC - STM32F4

DAC_DOR1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 – DACC1DOR[7:0]: Estos bits son de solo lectura, contienen datos de salida para DAC channel1.

DAC - STM32F4

DAC_DOR2:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 – DACC2DOR[7:0]: Estos bits son de solo lectura, contienen datos de salida para DAC channel2.

5.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	CEC EN	CAN2 EN	CAN1 EN	I2C4 EN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	SPDIFRX EN
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	Res.	LPTIM1 EN	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 clock enable
This bit is set and cleared by software.
0: UART8 clock disabled
1: UART8 clock enabled

Bit 30 **UART7EN**: UART7 clock enable
This bit is set and cleared by software.
0: UART7 clock disabled
1: UART7 clock enabled

Bit 29 **DACEN:** DAC interface clock enable
This bit is set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

```c
#include <stdio.h>
#include "stm32f7xx.h"

int SALIDA;
int tiempo;

extern "C"
{
  void SysTick_Handler(void)
  { tiempo++;
    if(tiempo == 1){
       GPIOB -> ODR = ~ GPIOB -> ODR;
      DAC -> DHR12R1 = DAC -> DHR12R1 + 400;
      tiempo = 0;
    }
  }
}
int main(void){

  RCC -> AHB1ENR |= 0X3; //PUERTO B
  RCC -> APB1ENR |= 0X20000000; //HABILITAR EL DAC
  GPIOA -> MODER |= 0Xf0;
  GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  DAC -> CR = 0X1;
  DAC -> DHR12R1 = 300;
  while(1){
      }
}
```

https://youtu.be/qiRQ9S0_ZFc

```c
#include "stm32f7xx.h"

int tiempo;

extern "C"
{
  void SysTick_Handler(void)
  { tiempo++;
    if(tiempo == 1){
       GPIOB -> ODR = ~ GPIOB -> ODR;
      DAC -> DHR12R1 = DAC -> DHR12R1 + 200;
      DAC -> DHR8R2 = DAC -> DHR8R2 + 200;
      tiempo = 0;
    }
  }
}
int main(void){
  RCC -> AHB1ENR |= 0X3; //PUERTO B
  RCC -> APB1ENR |= 0X20000000; //HABILITAR EL DAC
  GPIOA -> MODER |= 0Xf00;
  GPIOB -> MODER = 0X1; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  DAC -> CR = 0X10001;    //out 1-2
  DAC -> DHR12R1 = 500;
  DAC -> DHR8R2 = 500;
  while(1){     }
}
```

Ejercicio

Completar el siguiente código para que el microcontrolador genere un voltaje de 2 voltios en la salida del DAC (Vref=3.3V).

```
#include "stm32f7xx.h"

int main(void){
   RCC -> APB1ENR |= 0X20000000;
   DAC -> CR = 0X1;
   DAC -> DHR12L1 = 0X _____;
   while(1){
       }
}
```

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

SOLUCIÓN

```
//////////////   QUIZ  ////////////////////7
//  0X9B20=2V  0XC1F0=2,5V   0XE8B0=3V ///////

////////////////////////////////////////////////

#include "stm32f7xx.h"

int main(void){
   RCC -> APB1ENR |= 0X20000000;
   DAC -> CR = 0X1;
   DAC -> DHR12L1 = 0X9B20;
   while(1){
       }
}
```

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

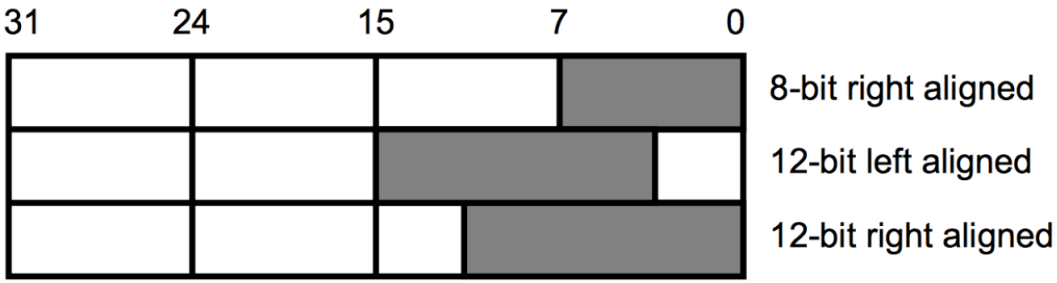
$$2V = \frac{3.3V \times DOR}{4096}$$

$$DOR = \frac{2V \times 4096}{3.3V} = 2482 = 0X9B2$$

DAC_DHR12L1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

0x	9	B	2	0



EJERCICIO DE CLASE

Variar la intensidad lumínica de un led a la salida de un canal DAC, mediante el uso de un potenciómetro en un canal de entrada análogo digital.

TAREA

Generar un tono audible que varié de frecuencia con el pulsador de la tarjeta, deben identificarse al menos 5 de ellas. Emplear un parlante o bocina de salida con el canal 1. Alinear a la izquierda.

Generar una señal de incremento/decremento de luz mediante un led. Utilizar pas
os de 10 incrementos y 10 decrementos para ir de mínimo a máximo y viceversa. Emplear el canal 2. Cada cambio debe realizarse a una frecuencia de 10 Hz