Using the DAC and DMA to generate a sine/sinc waveform

Hardware : SoC Stm32f4

References:
RM0090 Reference manual
AN3126 Application note

Contact : Mustapha.hamdi@insat.rnu.tn

Institut national des sciences appliquées et de technologie, Tunisia

Powered by STM32

Mustapha.hamdi@insat.rnu.tn

Powered by STM32

Introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

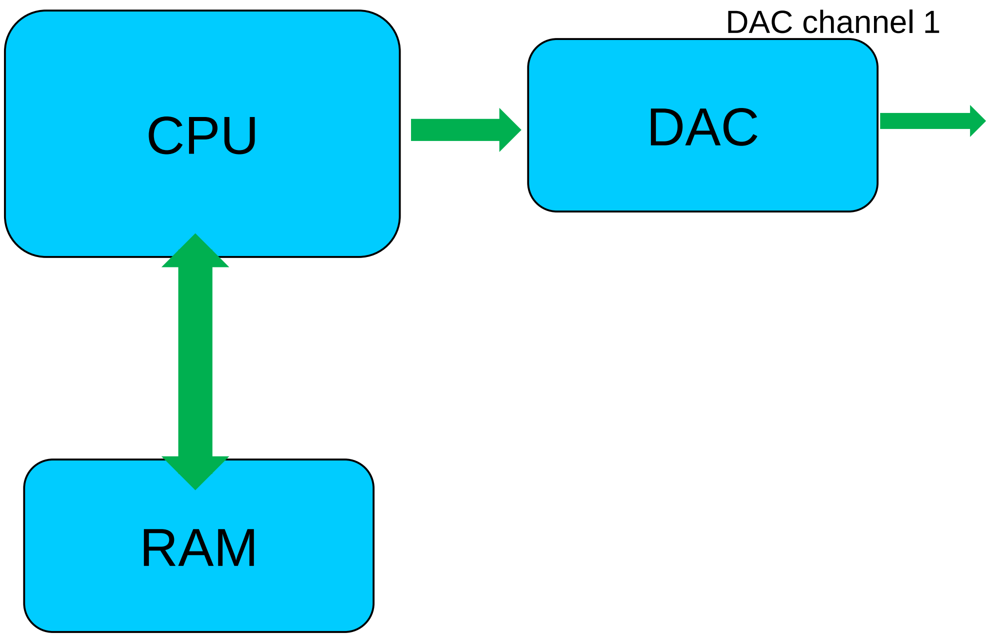
DMA

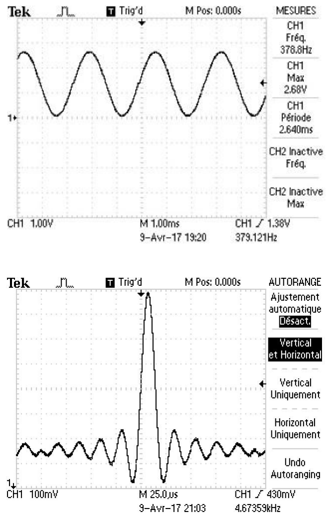
The STM32 microcontrollers have a DMA module with multiple channels. Each DAC channel is connected to an independent DMA channel. In the case of STM32F100x Microcontrollers, the DAC channel 1 is connected to the DMA channel 3 and DAC channel2 is connected to DMA channel 4.

When DMA is not utilized, the CPU is used to provide DAC with the pattern waveform. Generally the waveform is saved in a memory (RAM), and the CPU is in charge of transferring the data from RAM to the DAC.

DMA

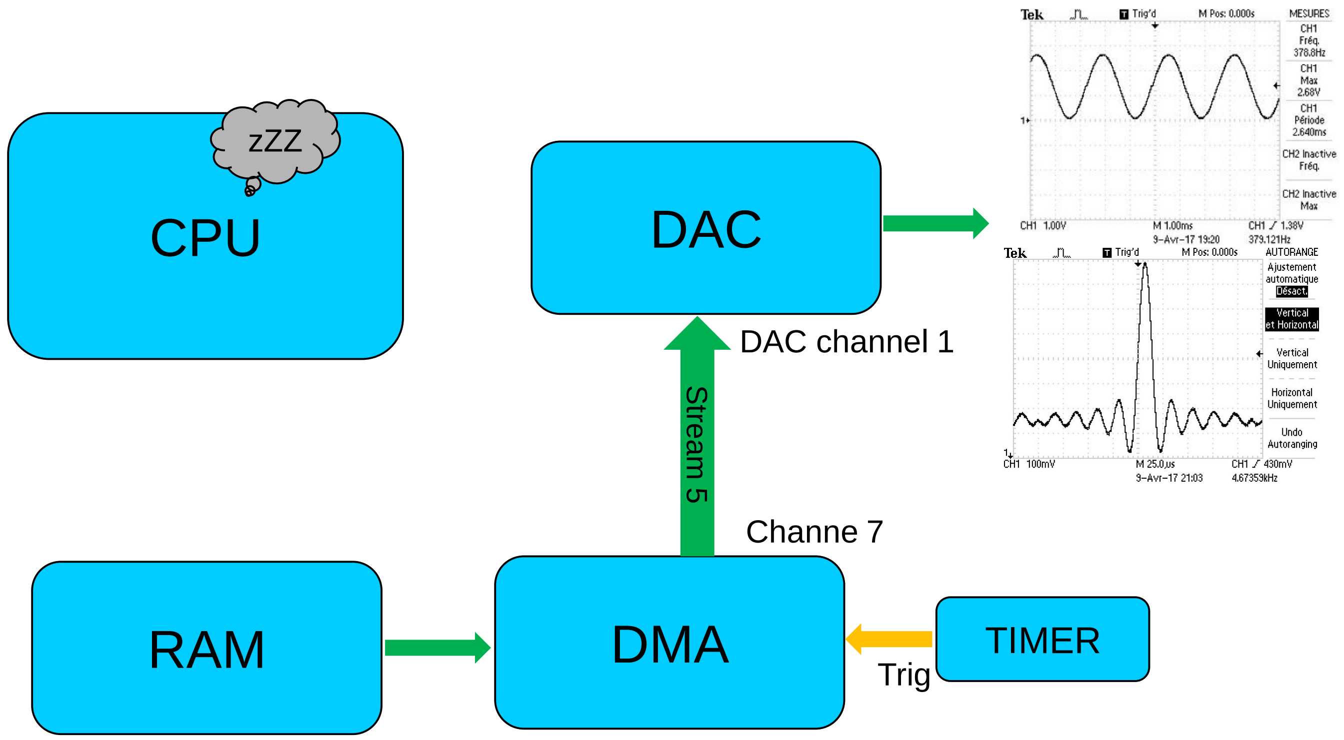
DAC interaction without DMA





Sinwave[DATA] = { 2048, 2145, 2242, 2339, 2435, 2530, 2624, 2717, 2808, 2897, 2984, 3069, 3151, 3230, 3307, 3381, 3451, 3518, 3581, 3640, 3696 …}

DAC interaction with DMA



Sinwave[resolution] = { 2048, 2145, 2242, 2339, 2435, 2530, 2624, 2717, 2808, 2897, 2984, 3069, 3151, 3230, 3307, 3381, 3451, 3518, 3581, 3640, 3696 …}

DMA

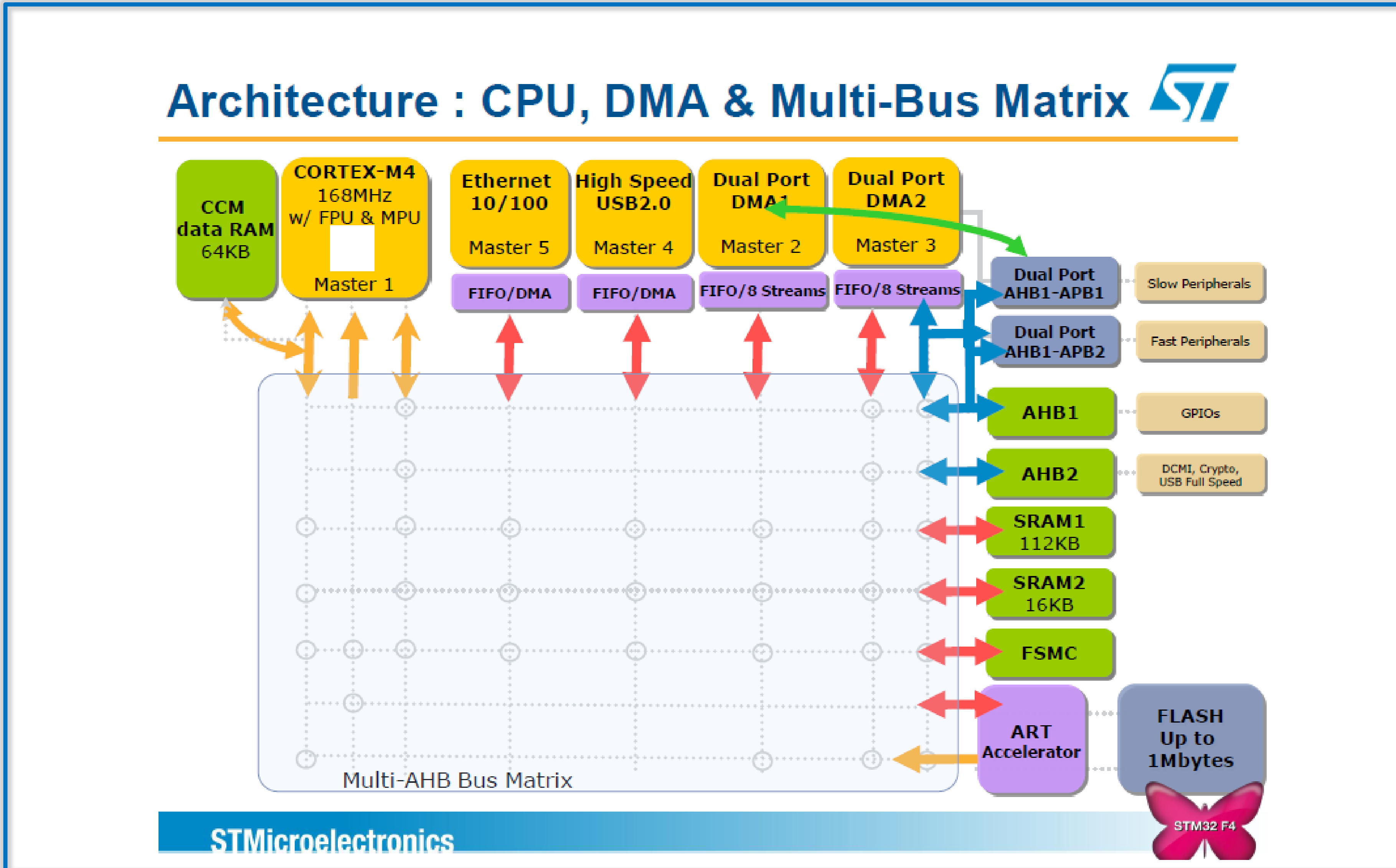
The **two DMA** controllers have **16 streams** in total (**8** for each controller), each dedicated to managing memory access requests from one or more peripherals. Each stream can have up to 8 channels (requests) in total.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The DMA controller performs direct memory transfer: as an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.
It can carry out the following transactions:

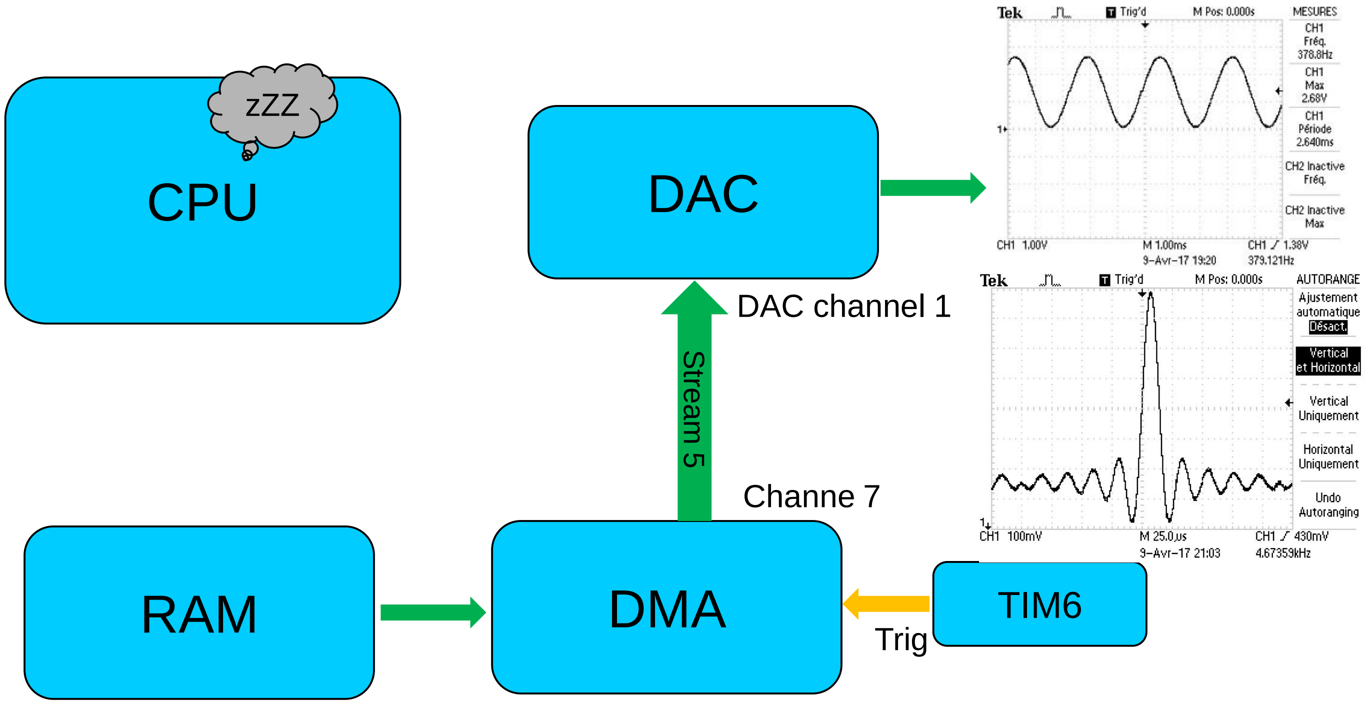
- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

NOC SoC smt32f4



DMA Use case

DAC interaction with DMA



Sinwave[DATA] = { 2048, 2145, 2242, 2339, 2435, 2530, 2624, 2717, 2808, 2897, 2984, 3069, 3151, 3230, 3307, 3381, 3451, 3518, 3581, 3640, 3696 …}

DMA

DMA configuration Registers:

Clock	AHBNER
DMA stream x configuration register	CR
Buffer size	NTDR
DMA stream x FIFO control register	FCR
peripheral adress	PAR
Memory adress	M0AR

Mustapha.hamdi@insat.rnu.tn

DMA

DMA configuration

DMA streams Data from memory to DAC peripheral via a specific channel, in our case, the DMA channel 7 is used to stream data to DAC Channel 1. It can be configured via stream x configuration register **(DMA_SxCR).**

```
DMA1_Stream5->CR |=0x7<<25;  // Channel 7 select
```

10.5.5 DMA stream x configuration register (DMA_SxCR) (x = 0..7)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[3:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM or reserved	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw or r	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **CHSEL[2:0]**: Channel selection

These bits are set and cleared by software.

000: channel 0 selected
001: channel 1 selected
010: channel 2 selected
011: channel 3 selected
100: channel 4 selected
101: channel 5 selected
110: channel 6 selected
111: channel 7 selected

These bits are protected and can be written only if EN is '0'

DMA

The direction is configured using the DIR[1:0] bits in the **DMA_SxCR** register and offers:

00: Peripheral-to-memory

01: Memory-to-peripheral

10: Memory-to-memory

11: reserved

```
DMA1_Stream5->CR |=0x1<<6;// Data transfer direction ,01: Memory-to-peripheral
```

DMA

Peripheral adress :

DAC adress

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC		*Section 14.5.15: DAC register map on page 450*
0x4000 7000 - 0x4000 73FF	PWR		*Section 5.6: PWR register map on page 146*
0x4000 6800 - 0x4000 6BFF	CAN2		

0x08	DAC_DHR12R1	Reserved	DACC1DHR[11:0]	
0x0C	DAC_DHR12L1	Reserved	DACC1DHR[11:0]	Reserved
0x10	DAC_DHR8R1	Reserved	DACC1DHR[7:0]	
0x14	DAC_DHR12R2	Reserved	DACC2DHR[11:0]	
0x18	DAC_DHR12L2	Reserved	DACC2DHR[11:0]	Reserved

DMA1_Stream5->PAR |= **0x40007408**; // peripheral adress 0x40007408

DMA

Circular mode

The Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

0: Circular mode disabled

1: Circular mode enabled

```
DMA1_Stream5->CR |=0x1<<8; //Circular mode
```

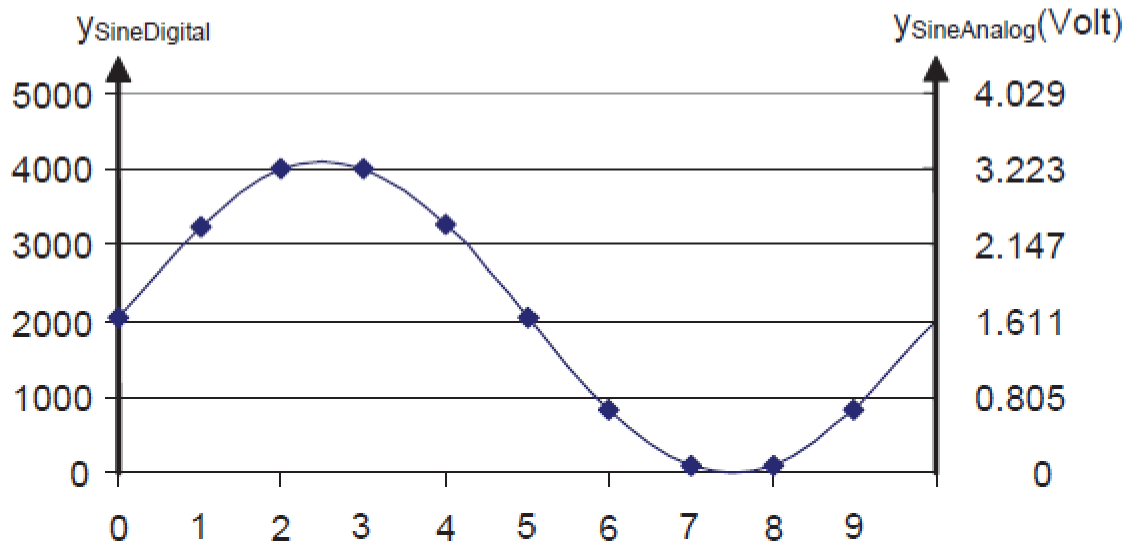
The buffer size is what tells the DMA how many bytes to transfer before wrapping around, and it will continue to loop through those addresses until explicitly disabled. It can be defined via the number of data register **NTDR**.

```
DMA1_Stream5->NDTR  |= 128 ; //number of data register :Buffer size
```

DMA

Digital Sine waveform preparation:

For example when our objective is to have 10 digital pattern data (samples) of a sine wave form which varies from 0 to 2*PI.



The sampling step is : (2*PI)/ ns (number of samples).

Because the result value of sin(x) is between -1 and 1, we have to recalibrate it to have a positive sinewave with samples varying between 0 and 0xFFF (which correspond, the range from 0 V to 3.3 V).

DMA

Sin Wave Frequency :

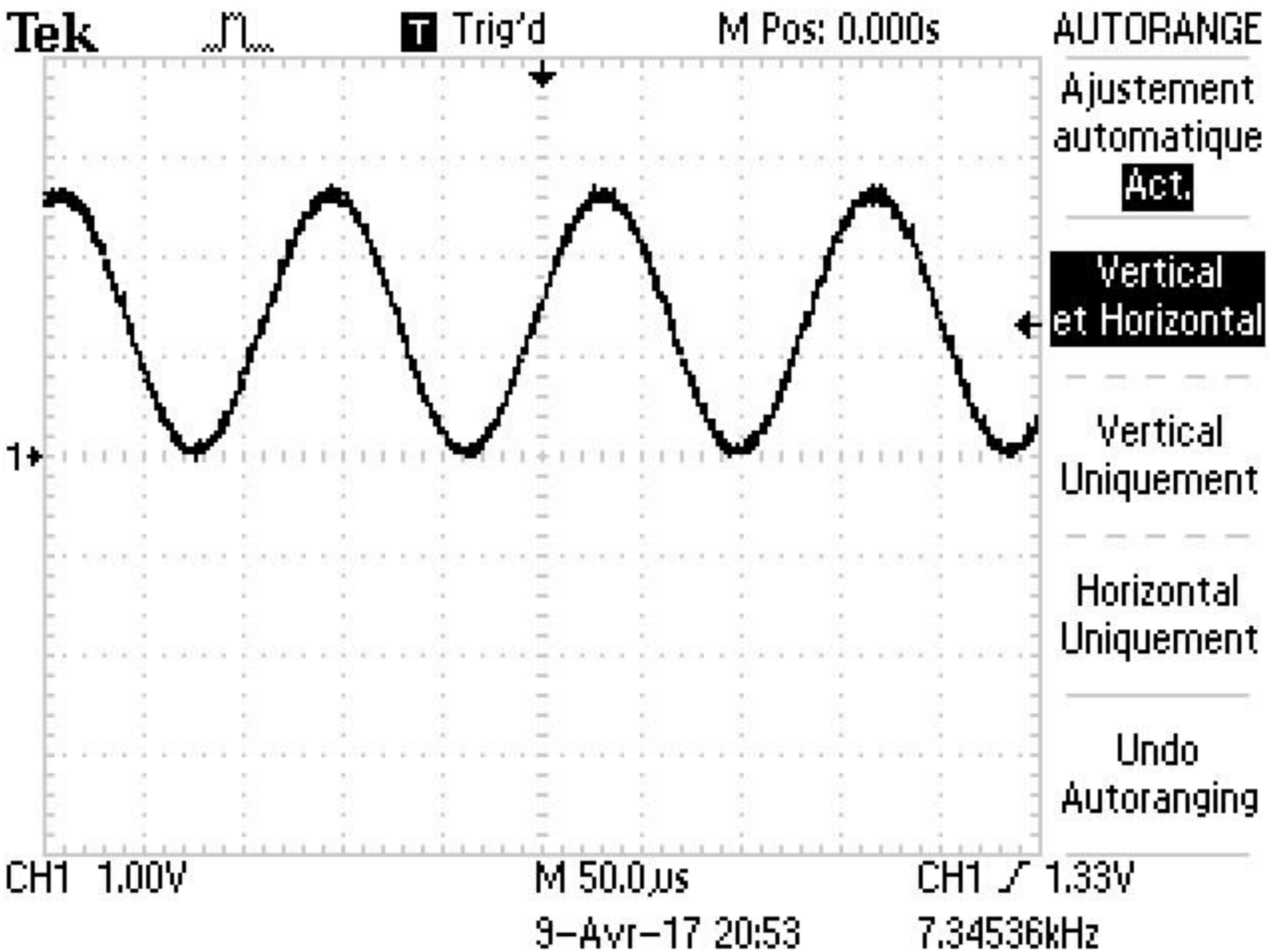
To fix the frequency of the sinewave signal, you have to set the frequency of the Timer Trigger output.

The frequency of the produced sine wave is

$f_{Sinewave} = f_{TimerTRGO}/ns$

So, if TIMx_TRGO is 1 MHz, the frequency of the DAC sine wave is 10 kHz.

Sin (x) function.



sinc(x) function.