1 TABLE DES MATIERES

2 EXPERIMENTATION AVEC LE DAC N°1 D'UNE CARTE NUCLEO-H755ZI

Ce tutoriel est une déclinaison du tutoriel « DAC-DMA-TIMER Nucléo-L476RG », mais il est plus direct, dans le sens où nous irons directement au but correspondant à l'étape 3.

La mise en œuvre du microcontrôleur STM32H755ZI est un véritable enfer ! Le résultat ici présenté est le fruit de nombreuses recherches et investigations que je passerai sous silence.

Défi : vous recherchez un bon passe-temps ? Si oui, ne lisez pas ce tutoriel et essayez de générer par vous-même un squelette de programme via STM32CubeMx dans le but de mettre en œuvre {DAC+TIMER2+DMA} et de créer une sinusoïde de fréquence variable.

La suite de ce tuto représente la solution à ce défi dans l'environnement MDK-ARM de µVision.

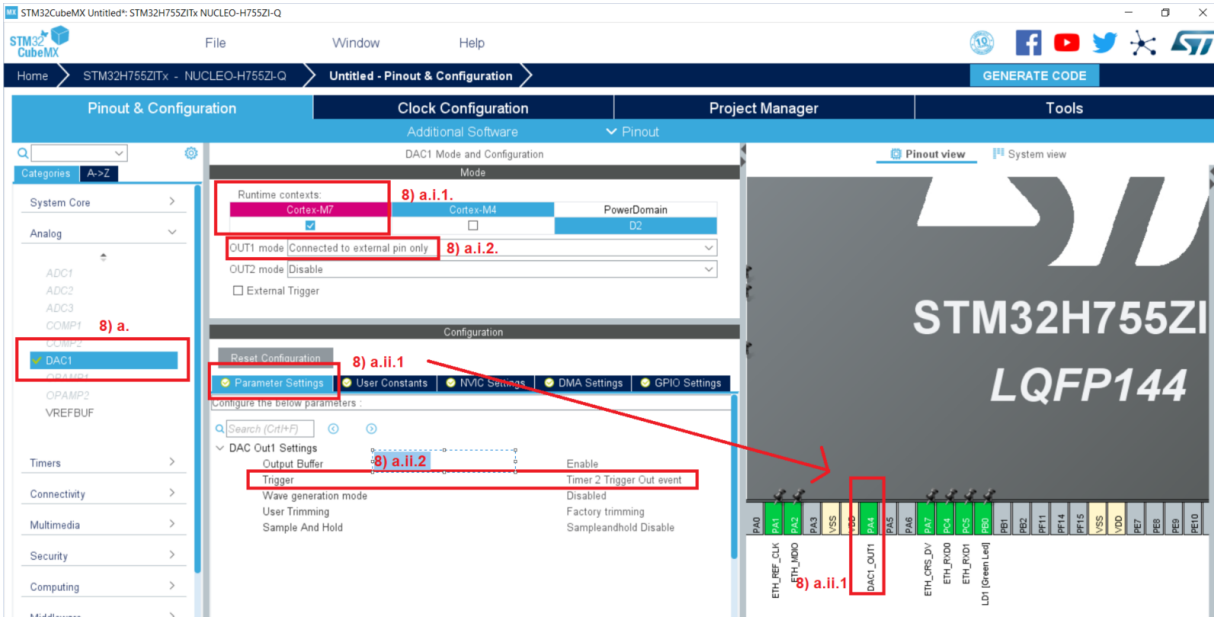
3 INITIALISATION DE L'ENVIRONNEMENT

STM32CubeMX peut grandement soulager l'utilisateur en générant le maximum de code à sa place, il suffit pour cela de lui indiquer nos choix en matière de :

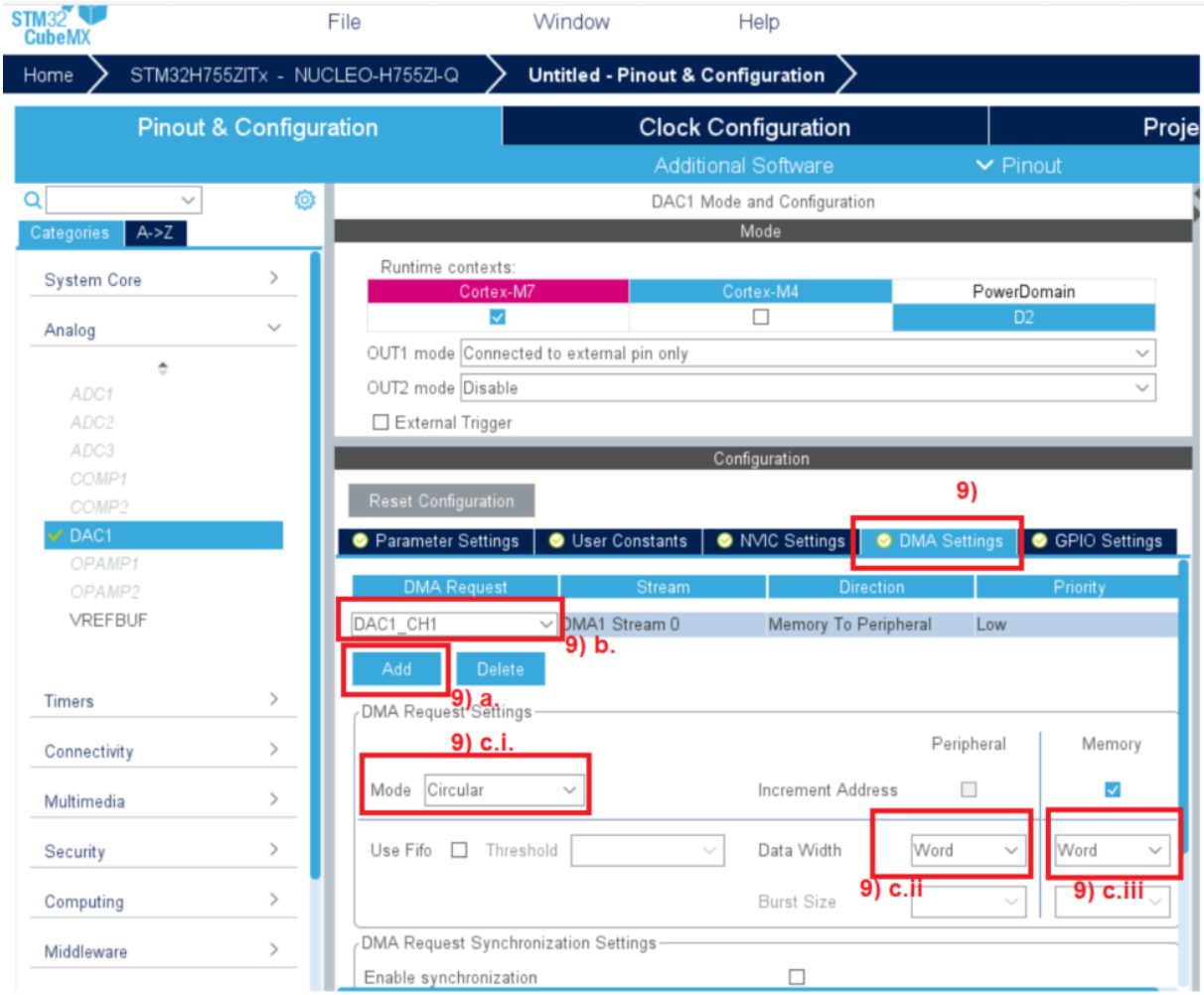
- carte Nucléo, ici ce sera le modèle Nucléo-H755ZI ;
- préférences pour la génération automatique de code ;
- périphérique à mettre en œuvre, ici ce sera le DAC1 ;
- fréquence d'horloge, on prendra la fréquence par défaut proposée par STM32CubeMX : 64 MHz ;
- nom et endroit de stockage du projet.
- Choix de l'IDE, ici ce sera MDK-ARM.

Voici comment lui indiquer ce que nous souhaitons :

1) Lancer STM32CubeIDE.
2) FILE/NewProject.
3) Choisir l'onglet « Board Selector »
4) Dans « Part Number Search », choisir NUCLEO-H755ZI-Q.
5) Cliquer dans la fenêtre de droite sur la carte Nucléo correspondante.
6) Cliquer sur le bouton « Start Project ».
7) « ***Initialize all peripherals with their default Mode*** » ➔ répondre **Yes**
8) Dans l'onglet « Pinout & Configuration »
    a. Développer Analog et cliquer sur DAC1
        i. Dans la partie Mode
            1. Dans Runtime contexts : cocher uniquement « Cortex-M7 »
            2. Out1 mode choisir Connected to external pin only.
        ii. Dans la partie Configuration
            1. Cliquer sur l'onglet Parameter Settings ➔ dans la partie pinout view vérifier que DAC1_OUT1 est assigné à PA4.
            2. Trigger choisir Timer 2 Trigger Out event.

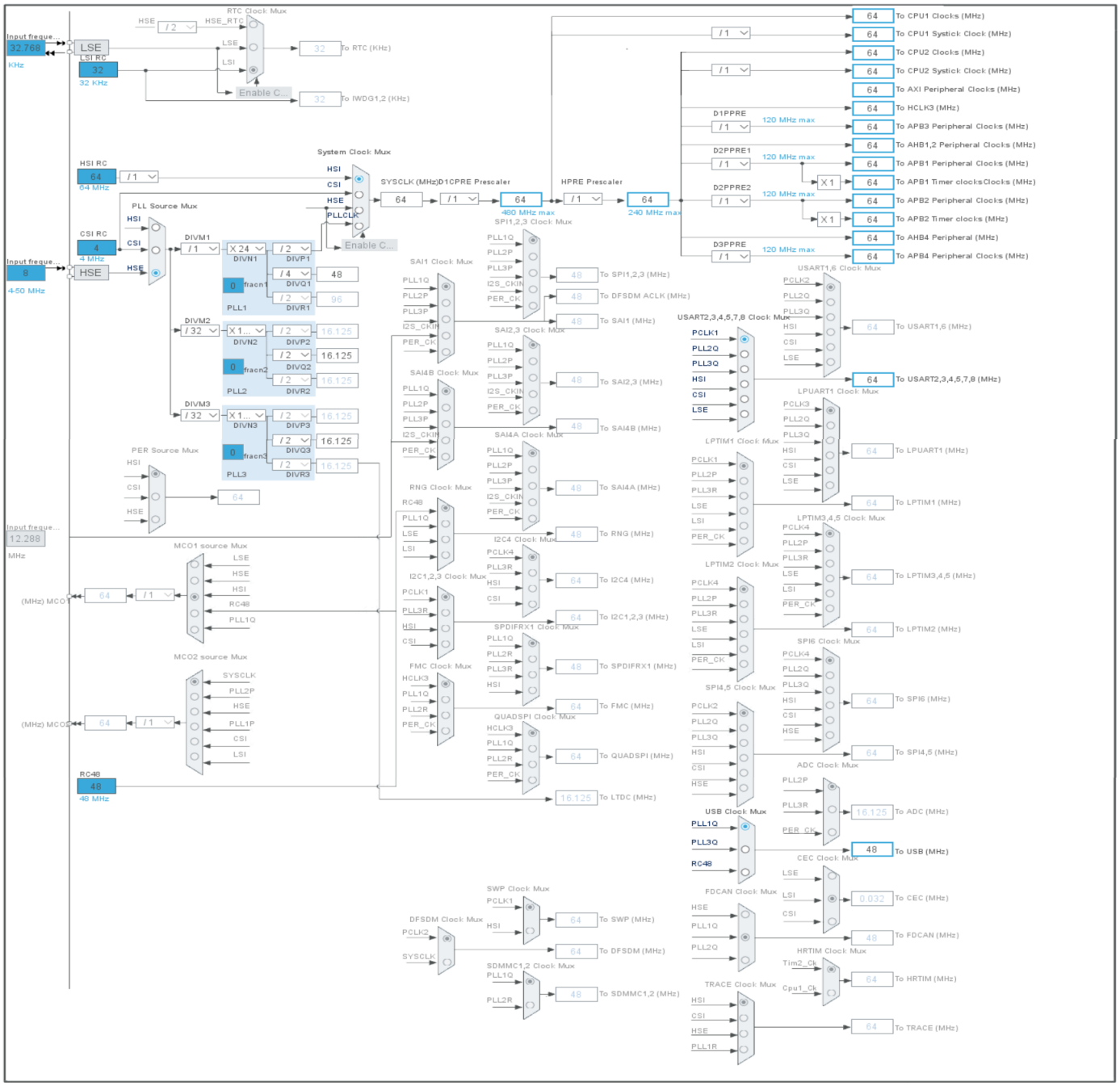


9) Partie « Configuration » du DAC1 », dans l'onglet « DMA settings » :
   a. Cliquer sur le bouton « Add » ;
   b. Dans la boîte « select » qui s'ouvre, choisir « DAC1_CH1 »
   c. Dans la partie « DMA Request Settings »
      i. choisir le Mode « Circular »
      ii. Data Width : « Word » (du côté Peripheral)
      iii. Data Width : « Word » (du côté Memory).



10) Onglet Clock Configuration
    a. Vérifier les fréquences à 64 MHz comme dans l'arbre d'horloge ci-dessous.

Pinout & Configuration | Clock Configuration | Project Manager | Tools



11) Il reste à configurer le timer2 que nous avons choisi :

a. Partie « Pinout & Configuration » ➔ choisir Tim2.

b. Partie « Tim2 Mode & Configuration /Mode », faire les réglages suivants :

   i. Dans Runtime contexts : cocher uniquement « Cortex-M7 »

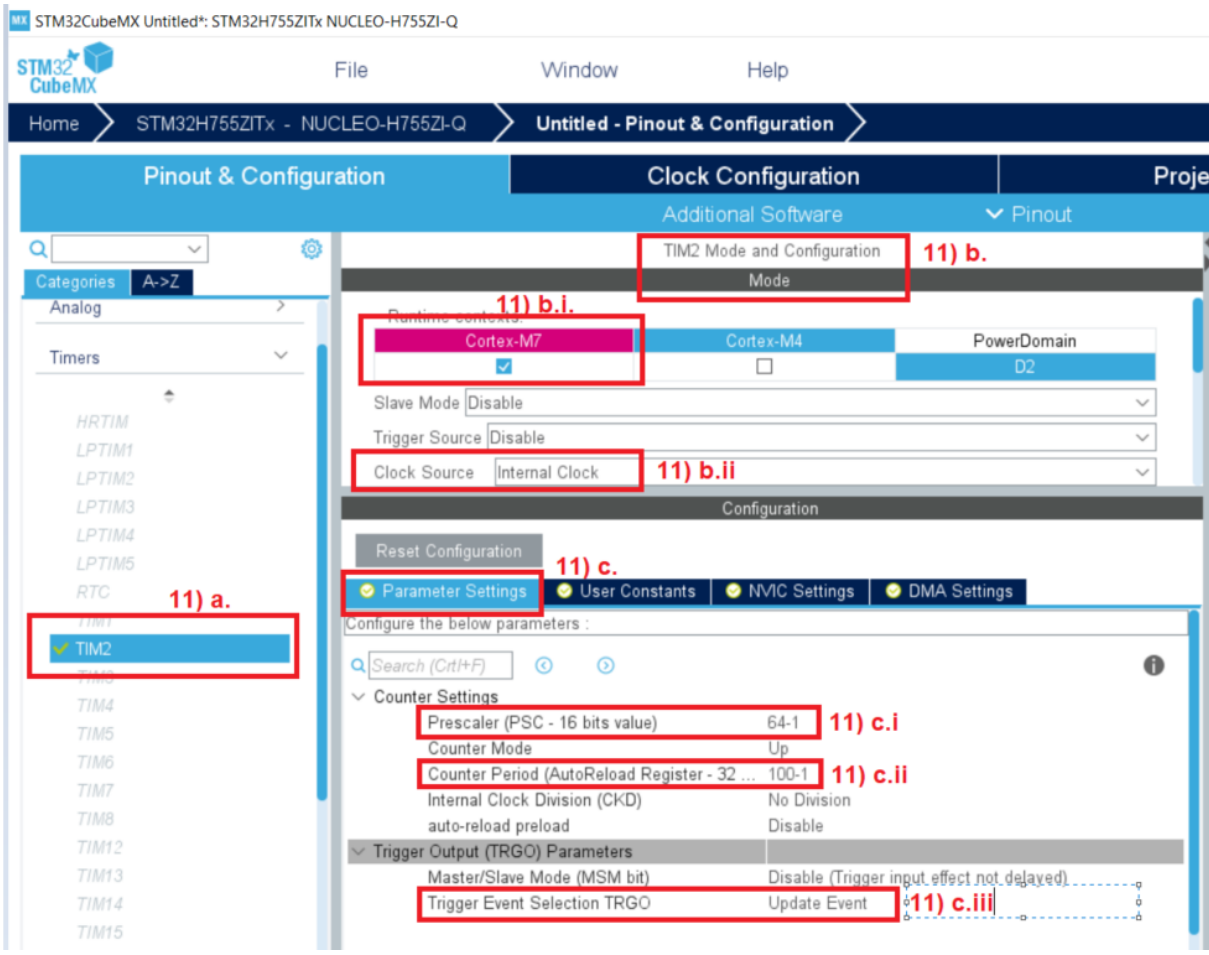
   ii. « Clock Source » ➔ « Internal Clock » ;

c. Partie « Tim2 Mode & Configuration /Configuration/Parameter Settings », faire les réglages suivants :

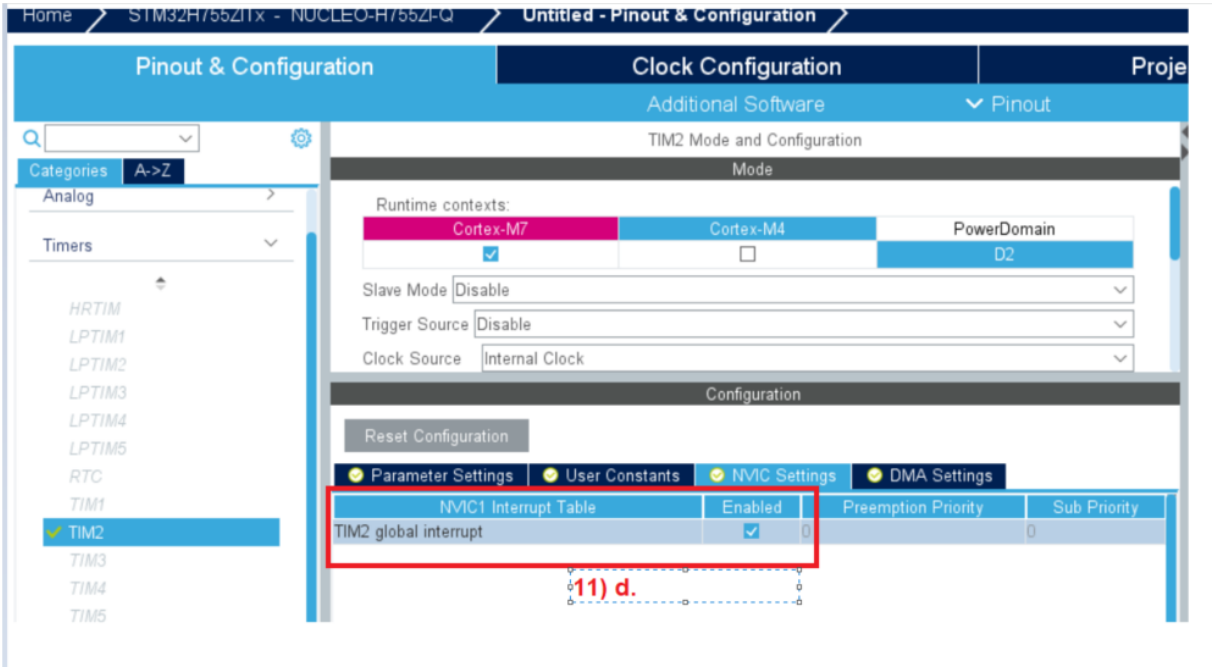
   i. Prescaler (PSC – 16 bit value) ➔ 64-1 Ceci a pour effet de diviser la fréquence d'horloge par 64, ce qui fait qu'en sortie du prescaler nous aurons 1 MHz en guise de fréquence d'horloge pour le timer2.

   ii. ➔ « Counter Period (AutoReload Register -32 bit value) » ➔ 100-1. Ceci a pour effet de générer une interruption tous les 100 coups d'horloge et donc d'obtenir une fréquence de 10 kHz pour le DMA.

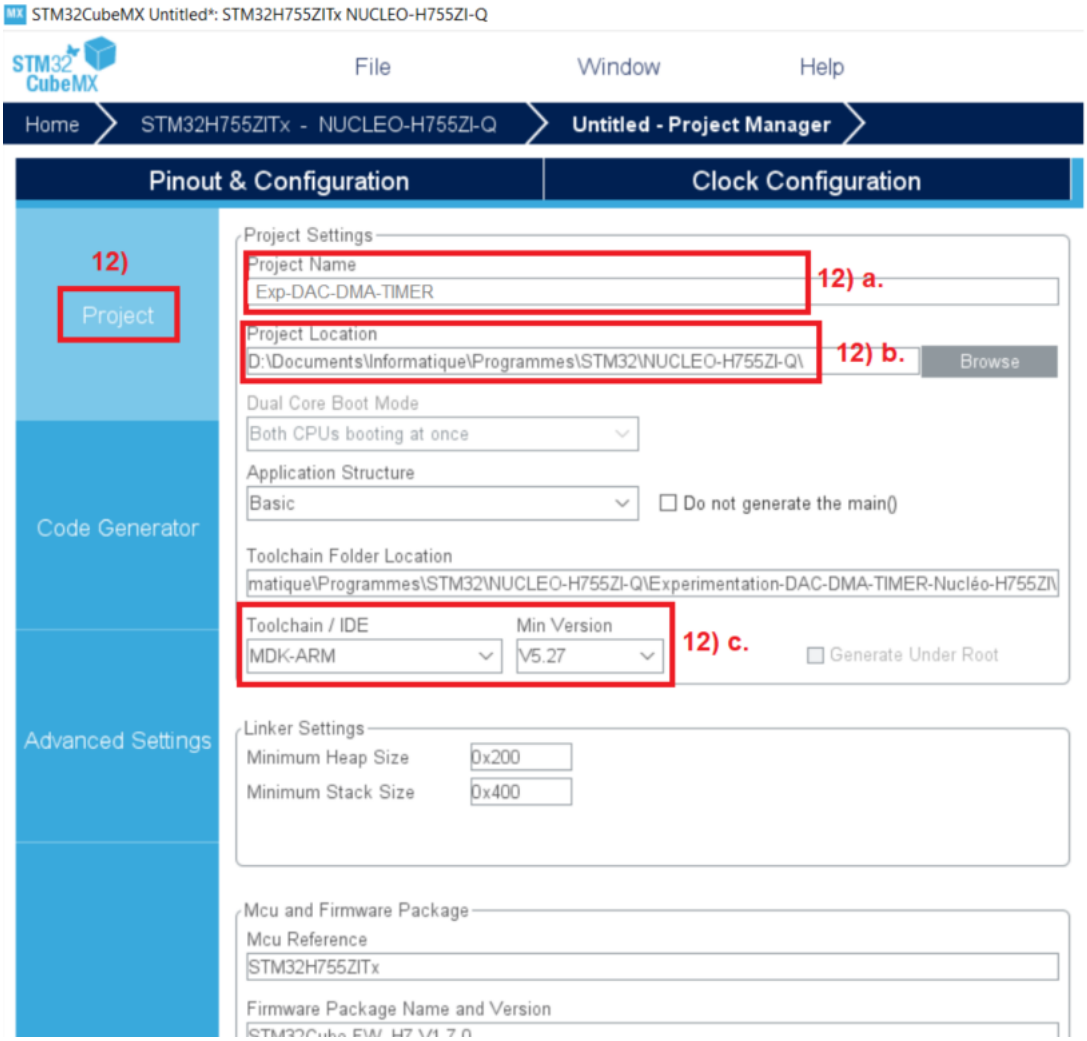
   iii. ➔ « Trigger Event Selection » ➔ « update event ».



d. ➔ Dans l'onglet « NVIC Settings » cocher « Enabled » en face de « TIM2 blobal interrupt ».

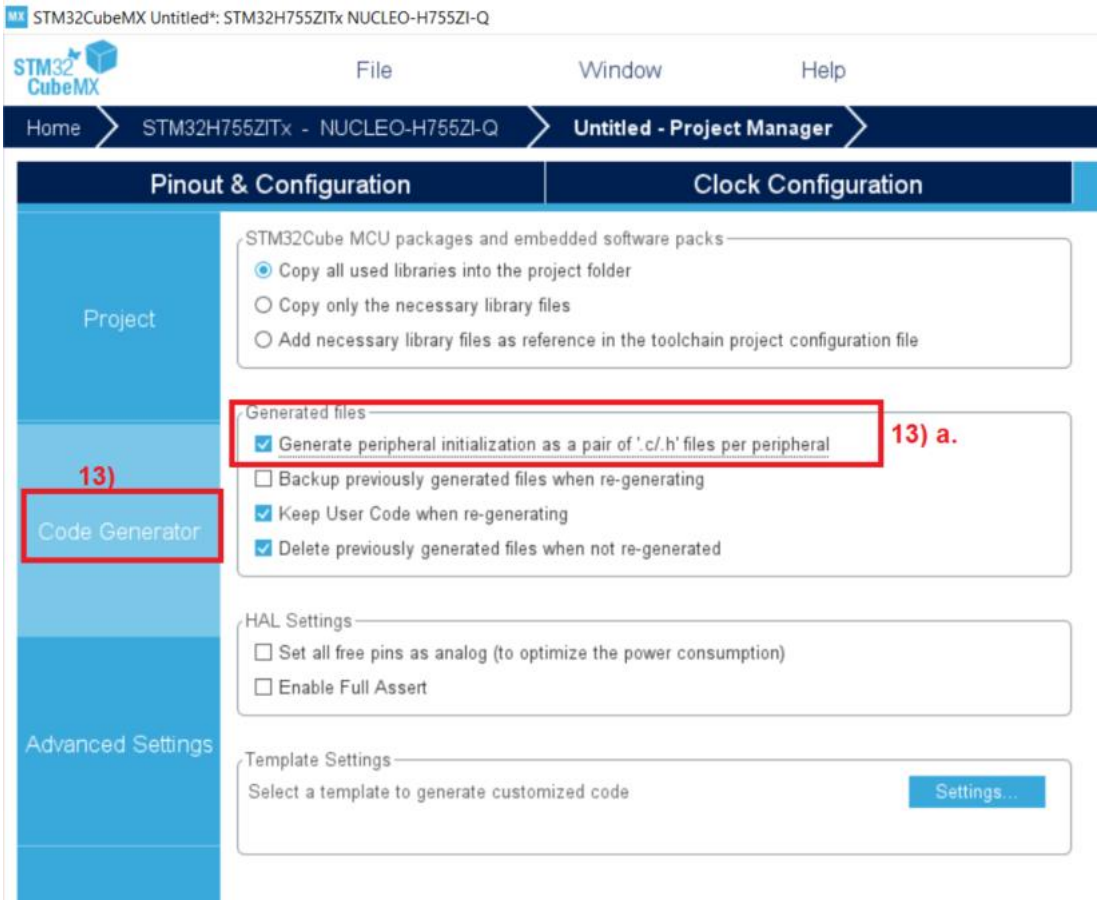


12) Onglet Project Manager/Project
    a. Nommer le projet
    b. Donner l'endroit où stocker le projet
    c. choisir l'IDE Toolchain/IDE: MDK-ARM pour ce qui me concerne.



13) Onglet Project Manager/Code Generator

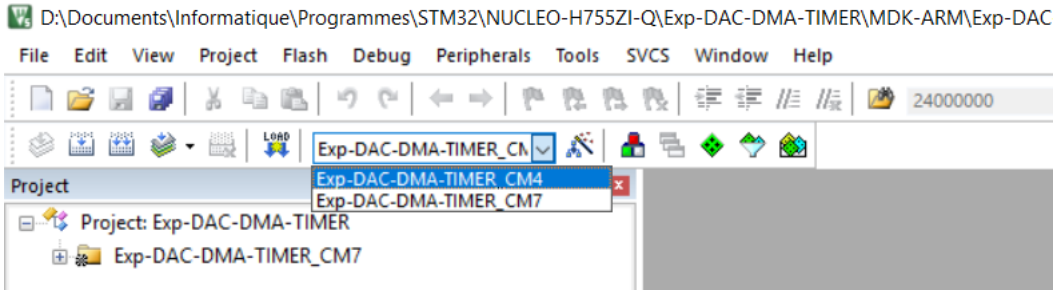
a. Onglet horizontal « Code Generator » ➔ cocher « *Generate peripheral initialization as a pair of '.c/h' files per peripheral* »



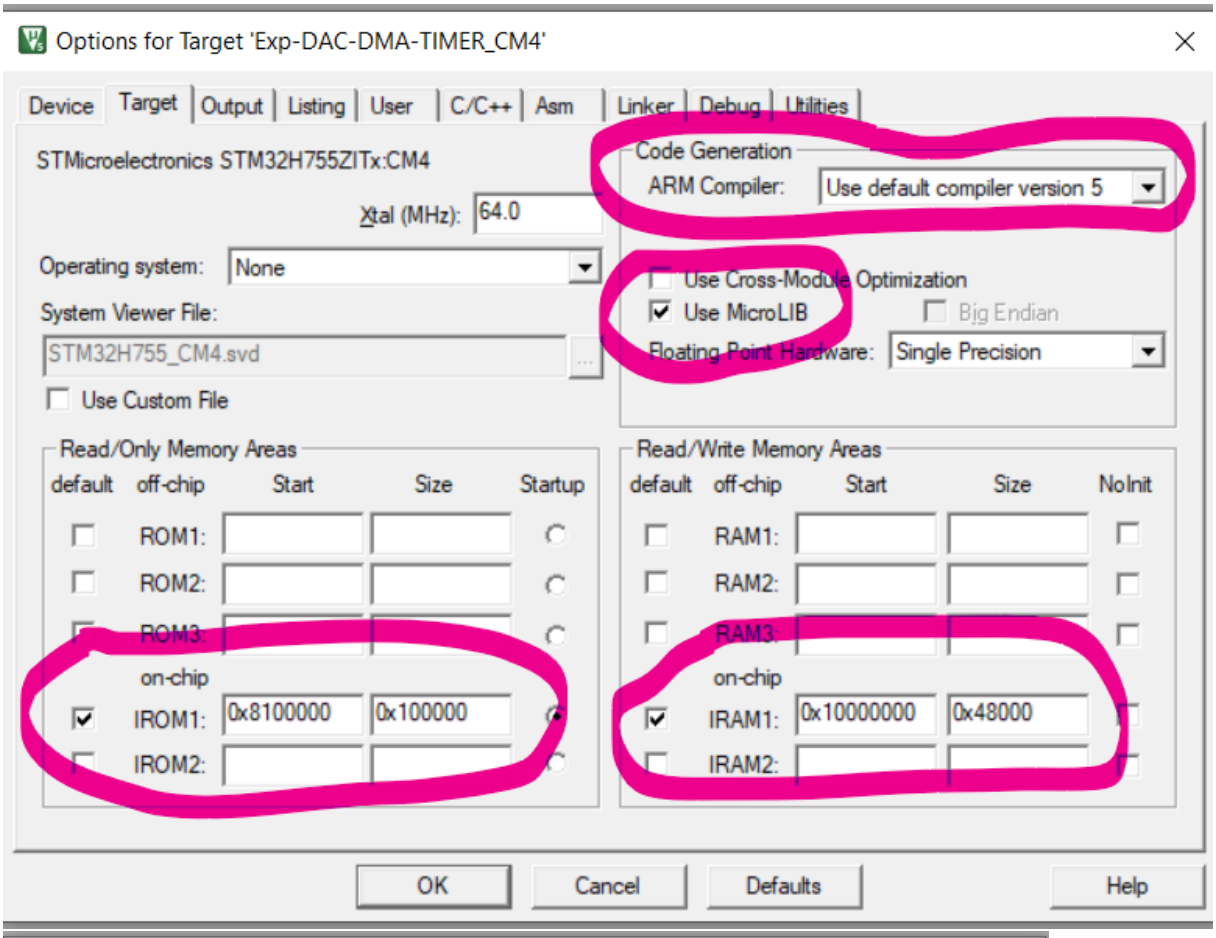
14) Fenêtre principale de STM32CubeMX
    a. Faire « FILE/SAVE »
    b. Appuyer sur le bouton Generate Code.

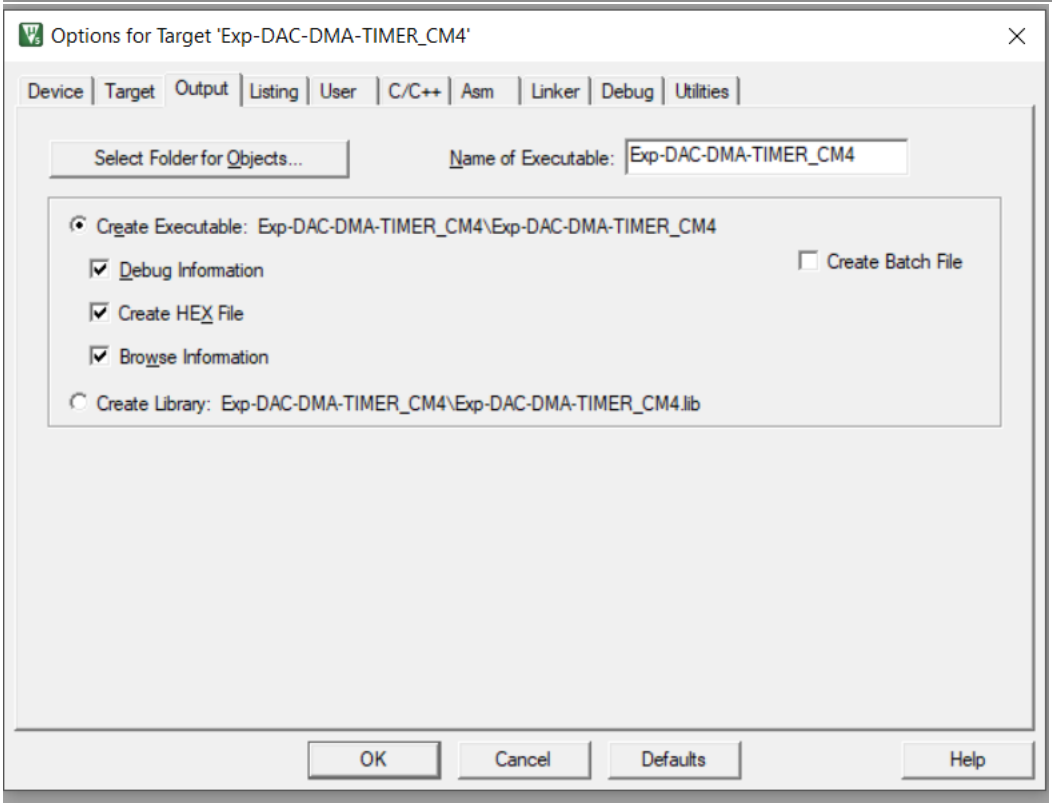
    c. Dans le popup qui s'affiche appuyer sur le bouton « Open Project »…

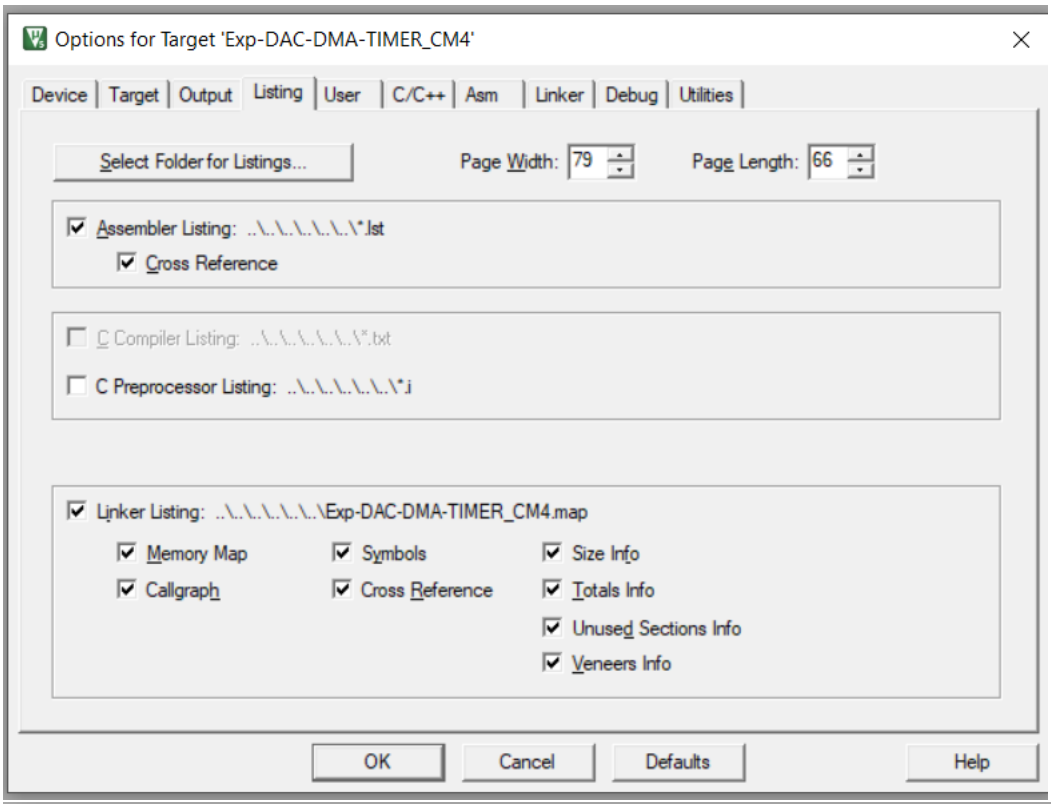
15) L'IDE Keil µVision s'ouvre, il faut changer les paramètres :
    a. Choisir d'abord la cible CM4

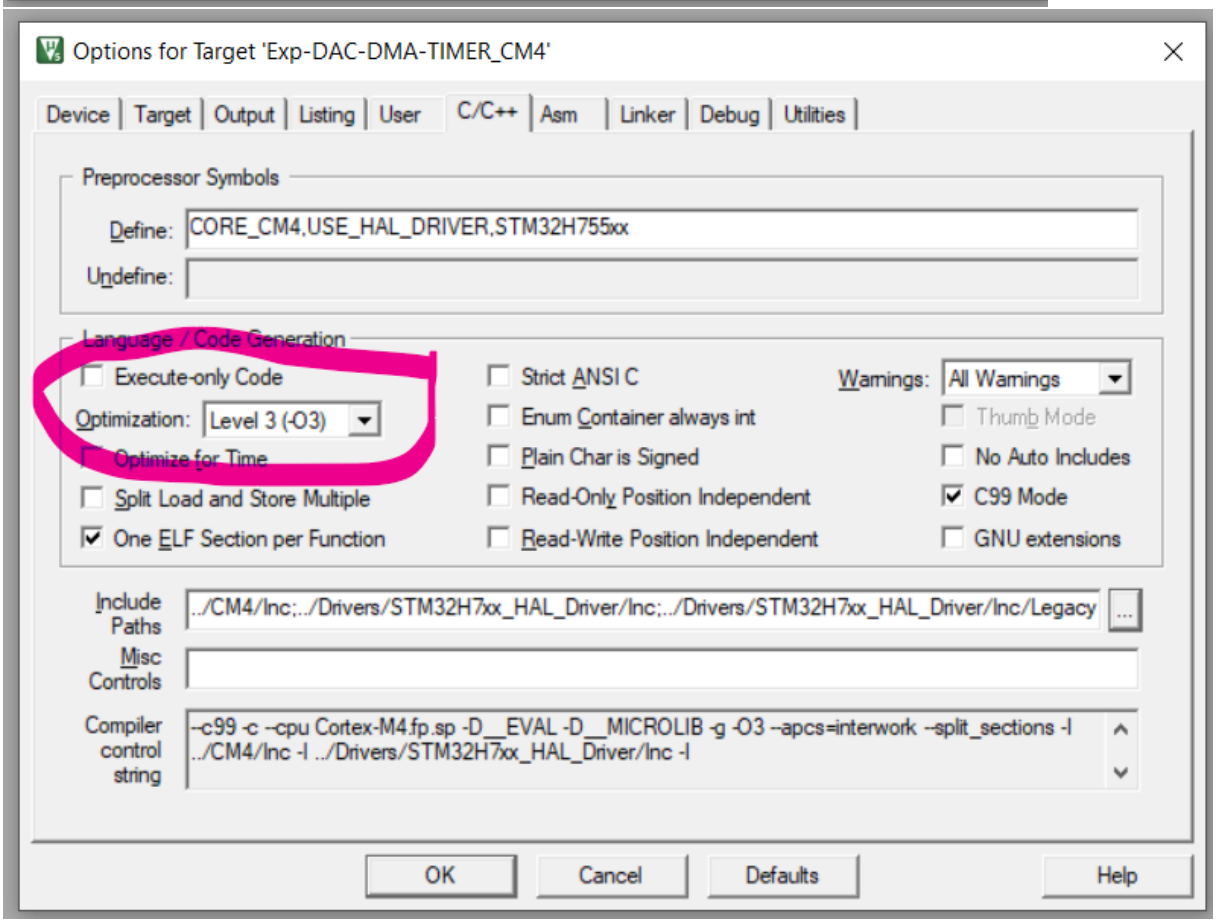


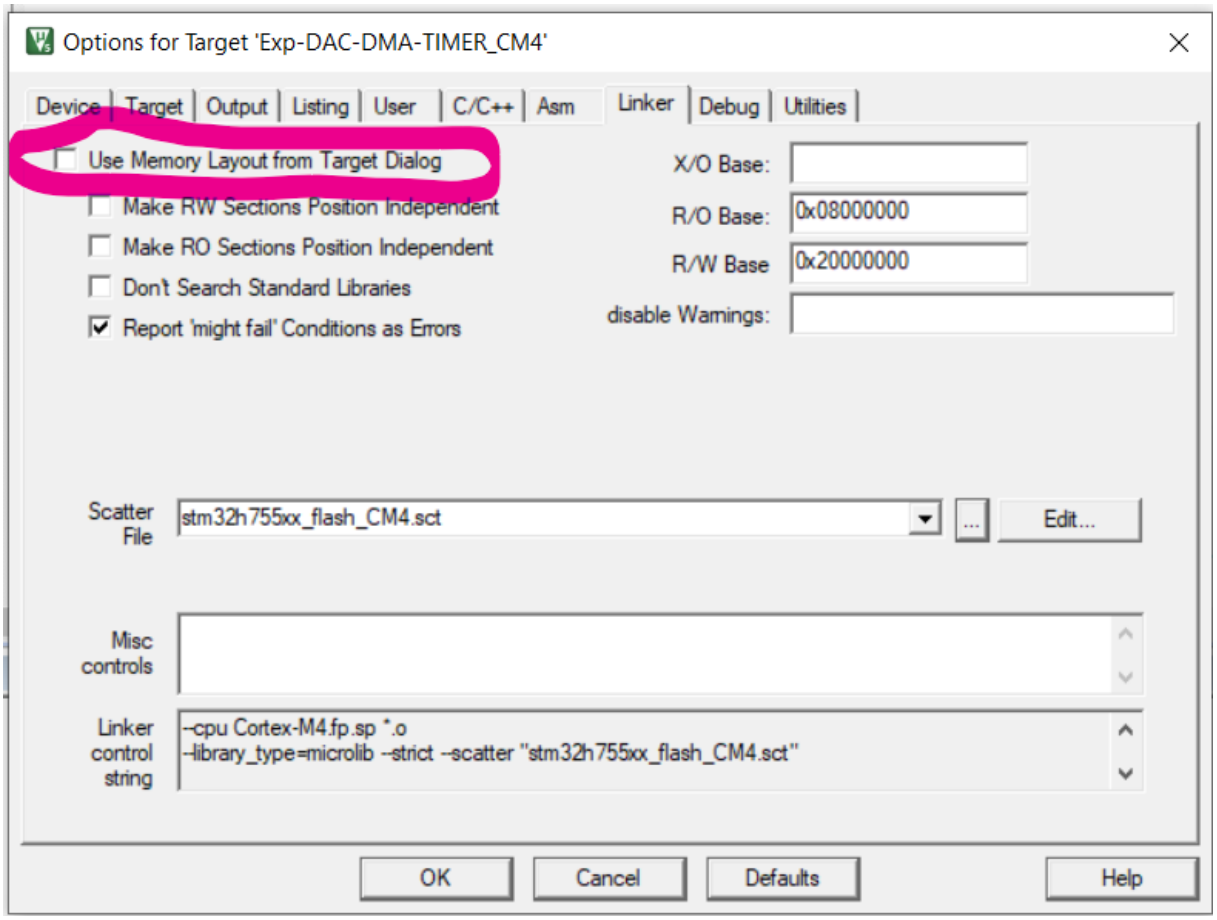
    b. Ouvrez les options du projet avec ALT+F7 et vérifiez scrupuleusement les écrans suivants :


Options for Target 'Exp-DAC-DMA-TIMER_CM4'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
STMicroelectronics STM32H755ZITx:CM4
Xtal (MHz): 64.0
Operating system: None
System Viewer File:
STM32H755_CM4.svd
Use Custom File
Code Generation
ARM Compiler: Use default compiler version 5
Use Cross-Module Optimization
Use MicroLIB
Big Endian
Floating Point Hardware: Single Precision
Read/Only Memory Areas
default off-chip Start Size Startup
ROM1:
ROM2:
ROM3:
on-chip
IROM1: 0x8100000 0x100000
IROM2:
Read/Write Memory Areas
default off-chip Start Size NoInit
RAM1:
RAM2:
RAM3:
on-chip
IRAM1: 0x10000000 0x48000
IRAM2:
OK
Cancel
Defaults
Help


Options for Target 'Exp-DAC-DMA-TIMER_CM4'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Select Folder for Objects...
Name of Executable: Exp-DAC-DMA-TIMER_CM4
Create Executable: Exp-DAC-DMA-TIMER_CM4\Exp-DAC-DMA-TIMER_CM4
Debug Information
Create HEX File
Browse Information
Create Batch File
Create Library: Exp-DAC-DMA-TIMER_CM4\Exp-DAC-DMA-TIMER_CM4.lib
OK
Cancel
Defaults
Help


Options for Target 'Exp-DAC-DMA-TIMER_CM4'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Select Folder for Listings...
Page Width: 79
Page Length: 66
Assembler Listing: ..\..\..\..\..\..*.lst
Cross Reference
C Compiler Listing: ..\..\..\..\..\..*.txt
C Preprocessor Listing: ..\..\..\..\..\..*.i
Linker Listing: ..\..\..\..\..\..\Exp-DAC-DMA-TIMER_CM4.map
Memory Map
Symbols
Size Info
Callgraph
Cross Reference
Totals Info
Unused Sections Info
Veneers Info
OK
Cancel
Defaults
Help


Options for Target 'Exp-DAC-DMA-TIMER_CM4'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Preprocessor Symbols
Define: CORE_CM4,USE_HAL_DRIVER,STM32H755xx
Undefine:
Language / Code Generation
Execute-only Code
Optimization: Level 3 (-O3)
Optimize for Time
Split Load and Store Multiple
One ELF Section per Function
Strict ANSI C
Enum Container always int
Plain Char is Signed
Read-Only Position Independent
Read-Write Position Independent
Warnings: All Warnings
Thumb Mode
No Auto Includes
C99 Mode
GNU extensions
Include Paths
../CM4/Inc;../Drivers/STM32H7xx_HAL_Driver/Inc;../Drivers/STM32H7xx_HAL_Driver/Inc/Legacy
Misc Controls
Compiler control string
--c99 -c --cpu Cortex-M4.fp.sp -D__EVAL -D__MICROLIB -g -O3 --apcs=interwork --split_sections -I ../CM4/Inc -I ../Drivers/STM32H7xx_HAL_Driver/Inc -I
OK
Cancel
Defaults
Help

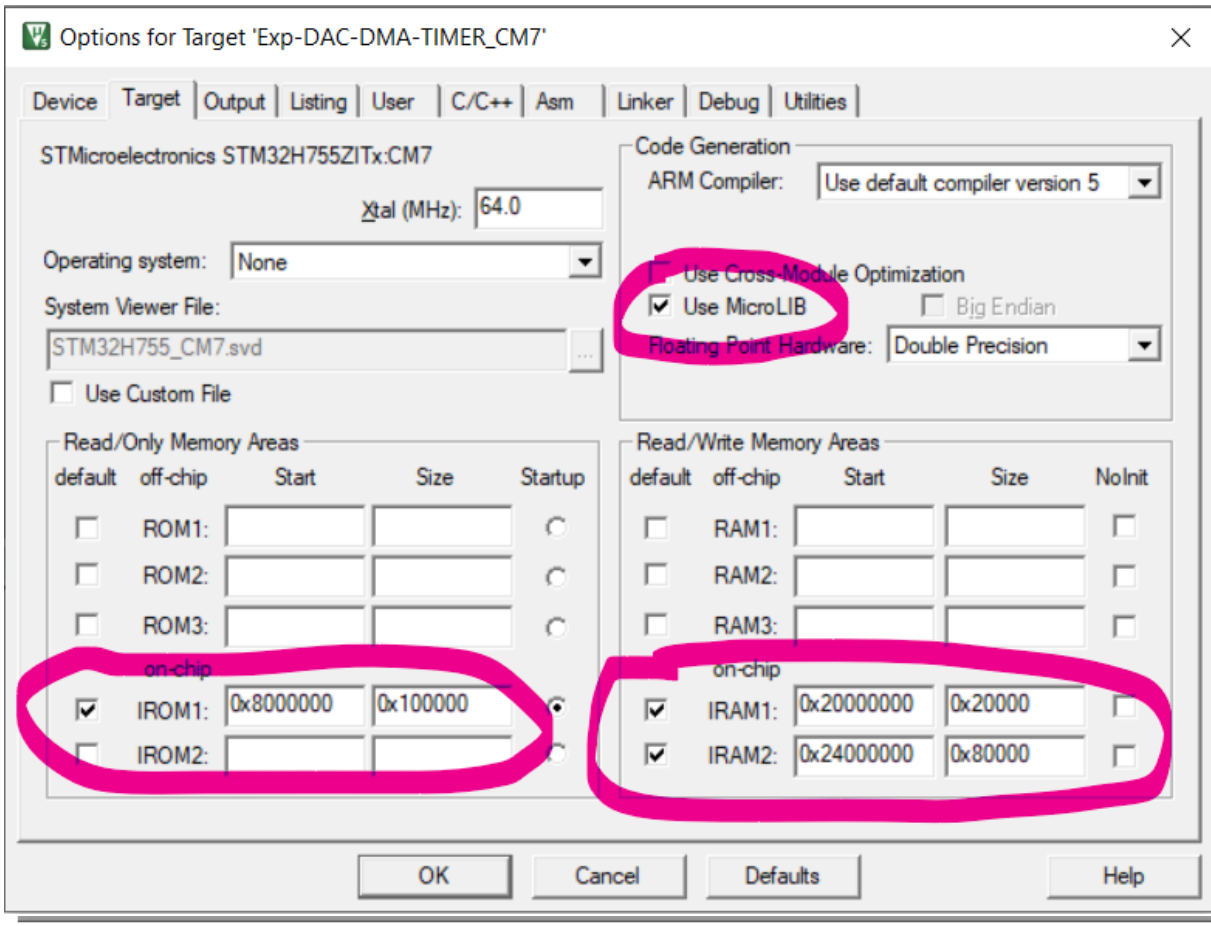


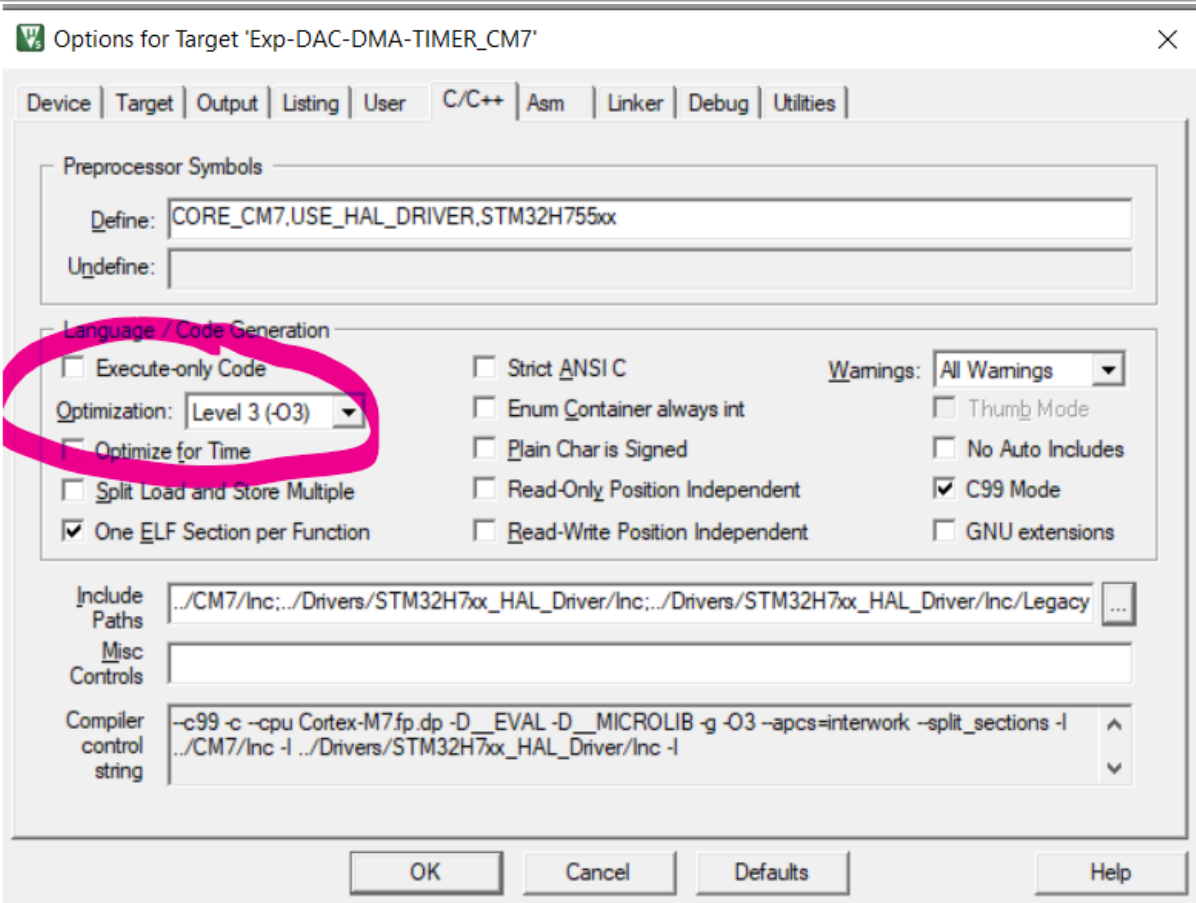
c. Compilez avec F7 et vous devriez obtenir ceci :

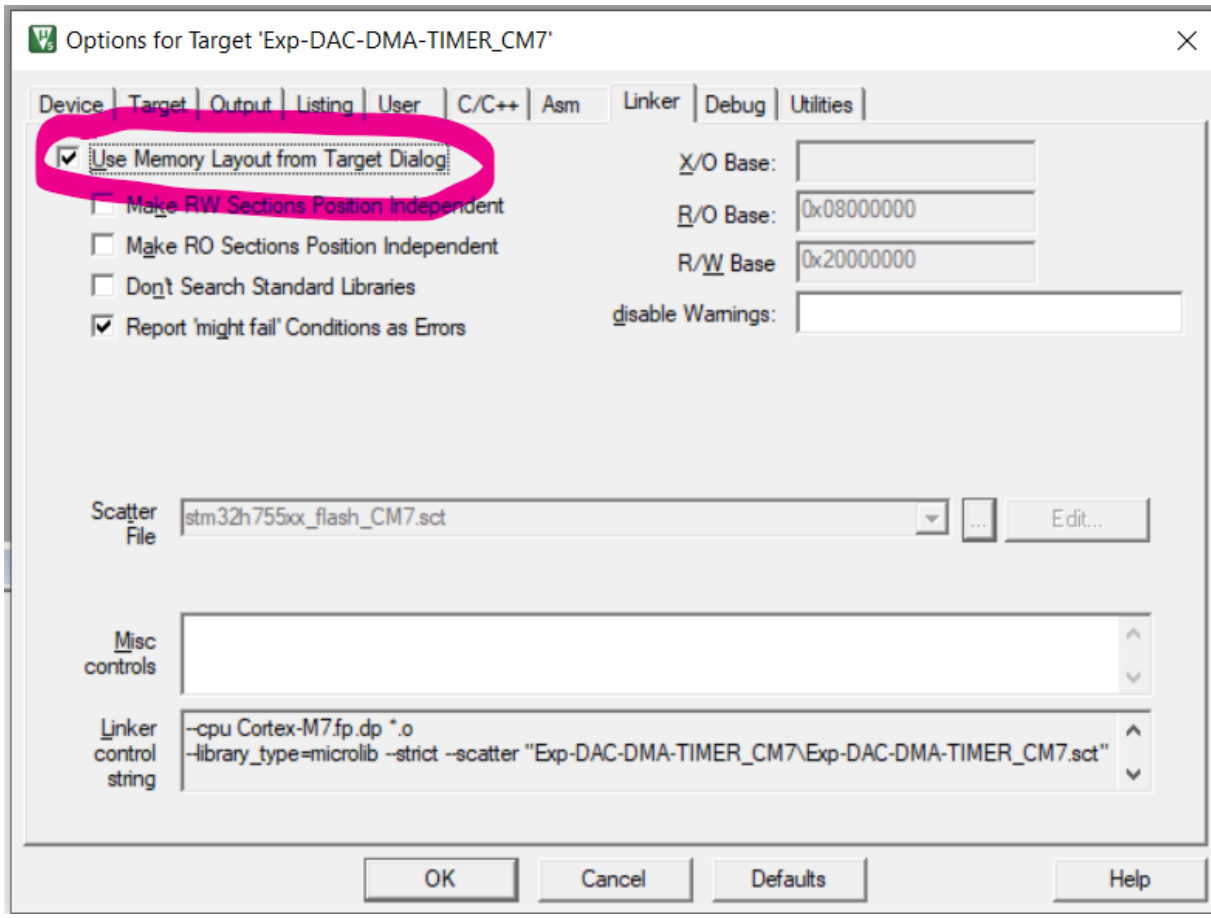
```
compiling stm32h7xx_hal_pwr_ex.c...
compiling stm32h7xx_hal_uart_ex.c...
compiling stm32h7xx_hal_tim_ex.c...
compiling system_stm32h7xx_dualcore_boot_cm4_cm7.c...
compiling stm32h7xx_hal_tim.c...
compiling stm32h7xx_hal_uart.c...
linking...
Program Size: Code=1200 RO-data=712 RW-data=20 ZI-data=7332
FromELF: creating hex file...
"Exp-DAC-DMA-TIMER_CM4\Exp-DAC-DMA-TIMER_CM4.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:02:20
```

d. Choisir maintenant la cible CM7

e. Ouvrez les options du projet avec ALT+F7 et vérifiez scrupuleusement les écrans suivants :


Options for Target 'Exp-DAC-DMA-TIMER_CM7'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
STMicroelectronics STM32H755ZITx:CM7
Xtal (MHz): 64.0
Operating system: None
System Viewer File:
STM32H755_CM7.svd
Use Custom File
Code Generation
ARM Compiler: Use default compiler version 5
Use Cross-Module Optimization
Use MicroLIB
Big Endian
Floating Point Hardware: Double Precision
Read/Only Memory Areas
default off-chip Start Size Startup
ROM1:
ROM2:
ROM3:
on-chip
IROM1: 0x8000000 0x100000
IROM2:
Read/Write Memory Areas
default off-chip Start Size NoInit
RAM1:
RAM2:
RAM3:
on-chip
IRAM1: 0x20000000 0x20000
IRAM2: 0x24000000 0x80000
OK
Cancel
Defaults
Help


Options for Target 'Exp-DAC-DMA-TIMER_CM7'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Preprocessor Symbols
Define: CORE_CM7,USE_HAL_DRIVER,STM32H755xx
Undefine:
Language / Code Generation
Execute-only Code
Optimization: Level 3 (-O3)
Optimize for Time
Split Load and Store Multiple
One ELF Section per Function
Strict ANSI C
Enum Container always int
Plain Char is Signed
Read-Only Position Independent
Read-Write Position Independent
Warnings: All Warnings
Thumb Mode
No Auto Includes
C99 Mode
GNU extensions
Include Paths ../CM7/Inc;../Drivers/STM32H7xx_HAL_Driver/Inc;../Drivers/STM32H7xx_HAL_Driver/Inc/Legacy
Misc Controls
Compiler control string --c99 -c --cpu Cortex-M7.fp.dp -D__EVAL -D__MICROLIB -g -O3 --apcs=interwork --split_sections -I ../CM7/Inc -I ../Drivers/STM32H7xx_HAL_Driver/Inc -I
OK
Cancel
Defaults
Help



f. Compilez avec F7 et vous devriez obtenir ceci :

```
compiling stm32h7xx_hal_pcd.c...
compiling system_stm32h7xx_dualcore_boot_cm4_cm7.c...
compiling stm32h7xx_ll_usb.c...
compiling stm32h7xx_hal_uart.c...
linking...
Program Size: Code=20344 RO-data=720 RW-data=20 ZI-data=268704204
"Exp-DAC-DMA-TIMER_CM7\Exp-DAC-DMA-TIMER_CM7.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:03:08
```

4 CONCEPTION ET CODAGE

4.1 CONCEPTION

Nous souhaitons créer une forme d'onde sinusoïdale.

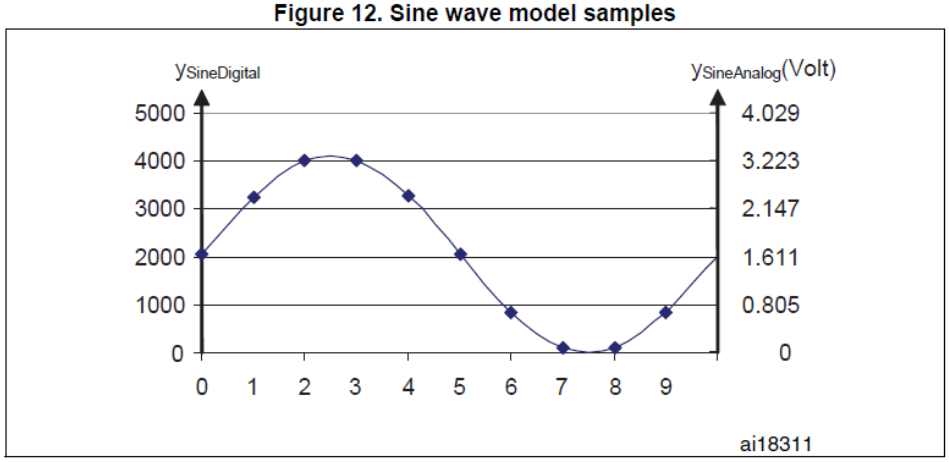
1) Ouvrir la note d'application fournie par ST : AN3126.
2) En 2.1.1 se trouve l'explication pas à pas de la méthode à suivre pour créer une forme sinusoidale.

2.1.2 Waveform preparation

To prepare the digital pattern of the waveform, we have to go through some mathematics.

Our objective is to have ten digital pattern data (samples) of a sine wave form that varies from 0 to 2π.

Figure 12. Sine wave model samples



The sampling step is 2π / n_s (number of samples).

The value of sin(x) varies between -1 and 1, we have to shift it up to have a positive sine wave with samples varying between 0 and 0xFFF (corresponding to the 0 to 3.3 V voltage range, where V_{REF} is set to 3.3 V).

$$y_{SineDigital}(x) = \left(\sin\left(x \cdot \frac{2\pi}{n_s}\right) + 1\right)\left(\frac{(0xFFF + 1)}{2}\right)$$

3) Page 16 on retrouve la formule permettant de calculer la sortie du DAC

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the equation

$$DAC_{Output} = V_{REF}\frac{DOR}{DAC\_MaxDigitalValue + 1}$$

Vref = 3,3 V

DOR est le registre dans lequel on met la valeur relative du signal à convertir en tension.

DAC_MaxDigitalValue dépend de la résolution que l'on utilise (12 bits ou 8 bits)

4.2 PROGRAMMATION

Ouvrir le fichier main.c du CM7 et réaliser ce qui est indiqué ci-dessous.

Inclure «math .h » :

```
31  /* Private includes ----------------------------------------------------------*/
32  /* USER CODE BEGIN Includes */
33  #include "math.h" // DAC1 : pour utilisation de la fonction sinus
34  /* USER CODE END Includes */
```

Ajouter un define pour la constante PI :

```
41  /* Private define ------------------------------------------------------------*/
42  /* USER CODE BEGIN PD */
43  #define PI 3.1415926 // DAC1 : pour utilisation de la fonction décrite dans AN3126
44  #define HSEM_ID_0 (0U) /* HW semaphore 0*/
45  /* USER CODE END PD */
```

Nous allons avoir besoin de quelques variables et là, il ne faut pas louper l'attribut qui sert à indiquer au linker où doit se placer le tableau dont nous aurons besoin !!! Ce n'est dit nulle-part mais le DMA associé au DAC ne fonctionne pas si l'on ne place pas la zone mémoire en 0x30000000 !!!

```
54  /* USER CODE BEGIN PV */
55  uint32_t sine_val[100] __attribute__((at(0x30000000)));
56  int n = 100 ;              // DAC1 : Nb échantillons
57  int i;
58  /* USER CODE END PV */
```

Il faut maintenant créer un tableau de 100 échantillons. Puisque nous aurons une interruption dont la fréquence sera de 10 kHz, chaque échantillon sera produit toutes les 100 µs et comme nous en aurons 100 par forme d'onde, la période de l'onde générée sera de 100*100µs = 10 ms, c'est dire que la fréquence de l'onde générée sera de 1/0,01 = **100 Hz**.

Ce que confirme d'ailleurs la note d'application :

2.1.3 Setting the sine wave frequency

To set the frequency of the sine wave signal, the user has to set the frequency ($f_{TimerTRGO}$) of the timer trigger output.

The frequency of the produced sine wave is

$$f_{Sinewave} = \frac{f_{TimerTRGO}}{n_s}$$

Nous aurons besoin de la constante **PI = 3,1415926**.

Nous créerons une fonction **get_sineval()** dont le but sera de remplir le tableau d'échantillons avec les valeurs définies par la formule de la note d'application AN3126 (cf plus haut).

$$y_{SineDigital}(x) = \left(\sin\left(x \cdot \frac{2\pi}{n_s}\right) + 1\right)\left(\frac{(0xFFF + 1)}{2}\right)$$

Dans notre cas, le nombre de samples **ns** sera de 100 , et 0xFFF correspond à la résolution du DAC sur 12 bits. 0xFFF+1 = 4096 en décimal et donc (0xFFF+1)/2=2048.

Dans la fonction main() il faudra

- démarrer le timer tim2 en appelant l'API HAL_TIM_Base_Start(&htim2) ;
- appeler la fonction get_sineval() qui servira une seule fois pour remplir la zone mémoire avec les 100 valeurs de samples.
- démarrer le DAC avec l'API spéciale liée au DMA : HAL_DAC_Start_DMA(&hdac1,DAC1_CHANNEL_1,sine_val,100,DAC_ALIGN_12B_R);

Dans cette API, on passe :

- en 3ème paramètre, le pointeur sur la zone où sont stockés les échantillons ;
- en 4ème paramètre, le nombre d'échantillons.

Il est temps de définir la fonction servant à remplir le tableau d'échantillons.

```
66  /* Private user code ---------------------------------------------------------*/
67  /* USER CODE BEGIN 0 */
68  void get_sineval() // DAC1 : usage unique pour valorisation des samples ; n = nombre d'échantillons
69  {// Cf formule note d'application AN3126
70    for(int i=0;i<n;i++)
71    {
72      sine_val[i]=((sin(i*2*PI/n)+1)*2048);
73    }
74  }
75  /* USER CODE END 0 */
```

Et il faut maintenant réaliser la première initialisation du tableau, puis démarrer le timer et enfin le DMA associé au DAC1.

```
140  /* USER CODE BEGIN 2 */
141  get_sineval();              // Onde sinusoïdale avec 100 échantillons (n initialisé à 100)
142  HAL_TIM_Base_Start(&htim2) ; // Démarrage du timer 2
143  HAL_DAC_Start_DMA(&hdac1,DAC1_CHANNEL_1,sine_val,100,DAC_ALIGN_12B_R); // Démarrage du DMA/DAC1, on passe en paramètre
144  /* USER CODE END 2 */
```

Il reste à :

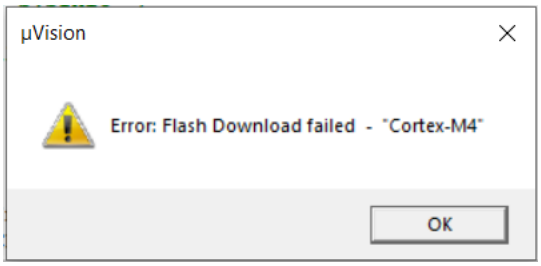
- compiler le main du CM7 via la touche F7 ;

```
Build Output
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'D:\Programmes\Keil_v5\ARM\ARMCC\Bin'
Build target 'Exp-DAC-DMA-TIMER_CM7'
compiling main.c...
linking...
Program Size: Code=23744 RO-data=1008 RW-data=28 ZI-data=268704196
"Exp-DAC-DMA-TIMER_CM7\Exp-DAC-DMA-TIMER_CM7.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:12
```

- charger le programme CM7 dans le STM32H755ZI via la touche F8 ;

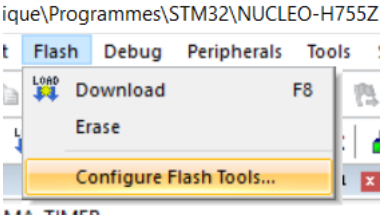
```
Build started: Project: Exp-DAC-DMA-TIMER
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'D:\Programmes\Keil_v5\ARM\ARMCC\Bin'
Build target 'Exp-DAC-DMA-TIMER_CM7'
compiling main.c...
linking...
Program Size: Code=23744 RO-data=1008 RW-data=28 ZI-data=268704196
"Exp-DAC-DMA-TIMER_CM7\Exp-DAC-DMA-TIMER_CM7.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:12
Load "Exp-DAC-DMA-TIMER_CM7\\Exp-DAC-DMA-TIMER_CM7.axf"
*** error 129: MapMem - map size truncated to 128MB
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 11:15:18
```

- De nouveau sélectionner la cible CM4 comme réalisé au chapitre 3, opération 15) a.
- Tenter de charger le programme CM7 dans le STM32H755ZI via la touche F8…

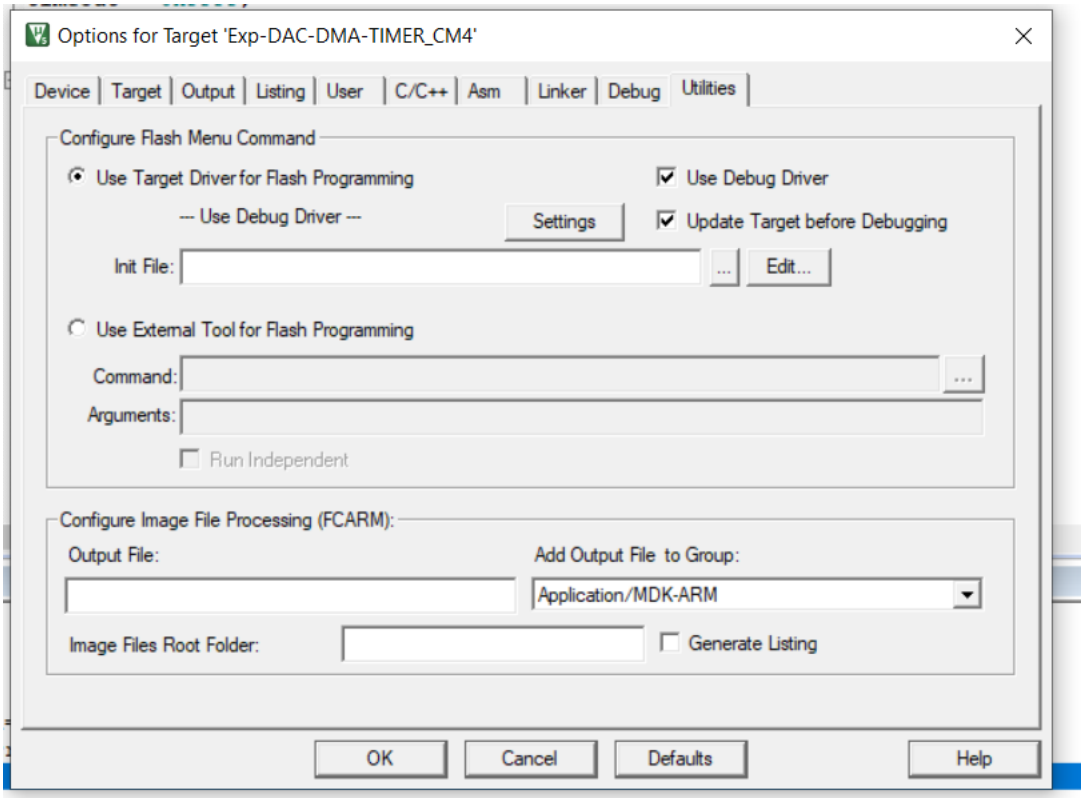


Selon l'état de développement de STM32CubeMX au moment où vous exécuterez ce tutoriel, vous aurez ou pas les messages d'erreur ci-dessus…

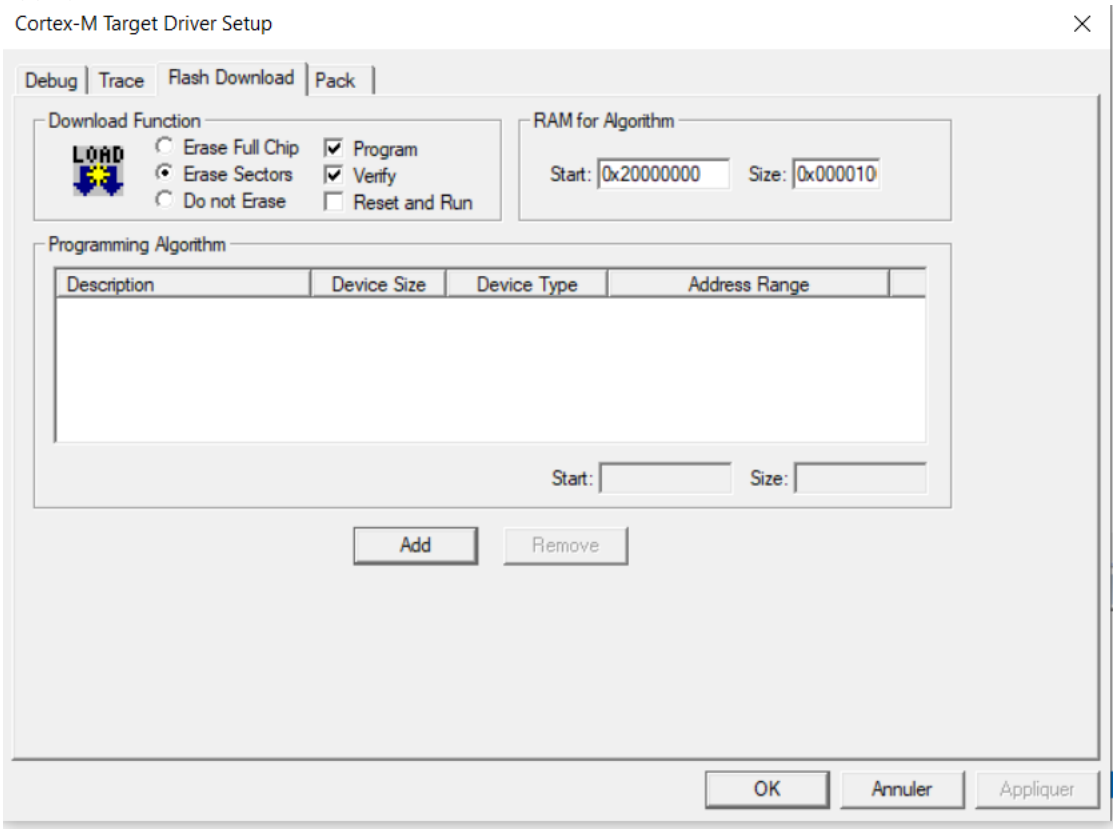
Pour en venir à bout, voici ce qu'il faut faire :

1) 

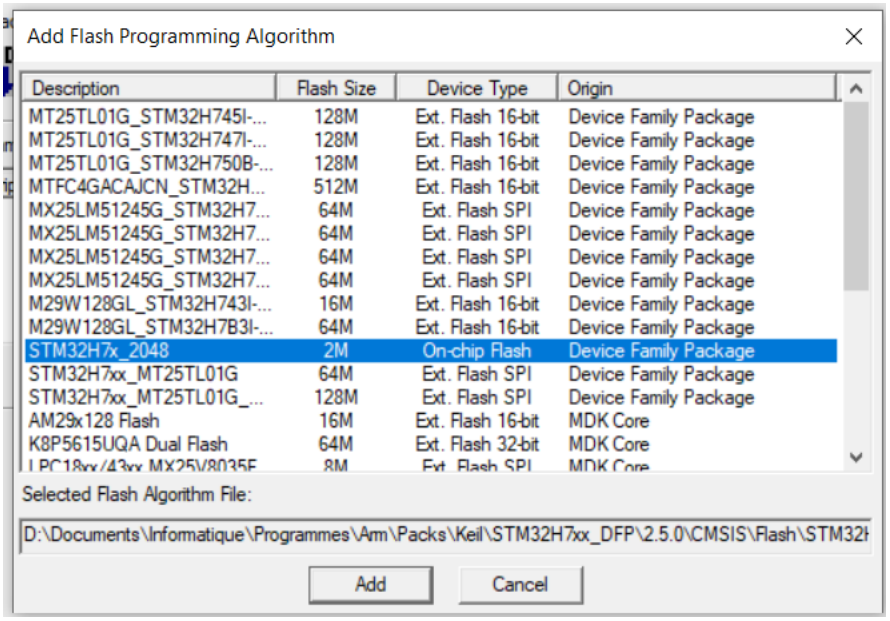
2) Appuyer sur Settings



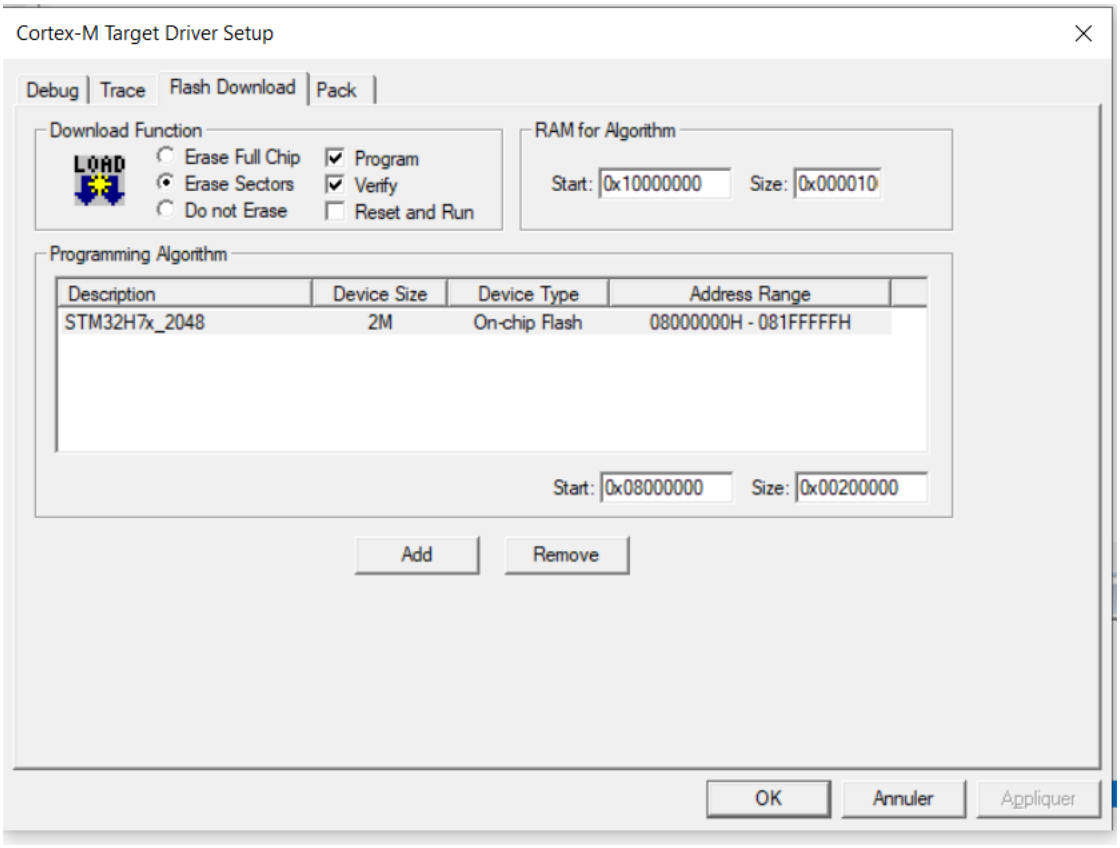
3) Appuyer sur Add



4) Choisir STM32H7x_2048 et appuyer sur ADD

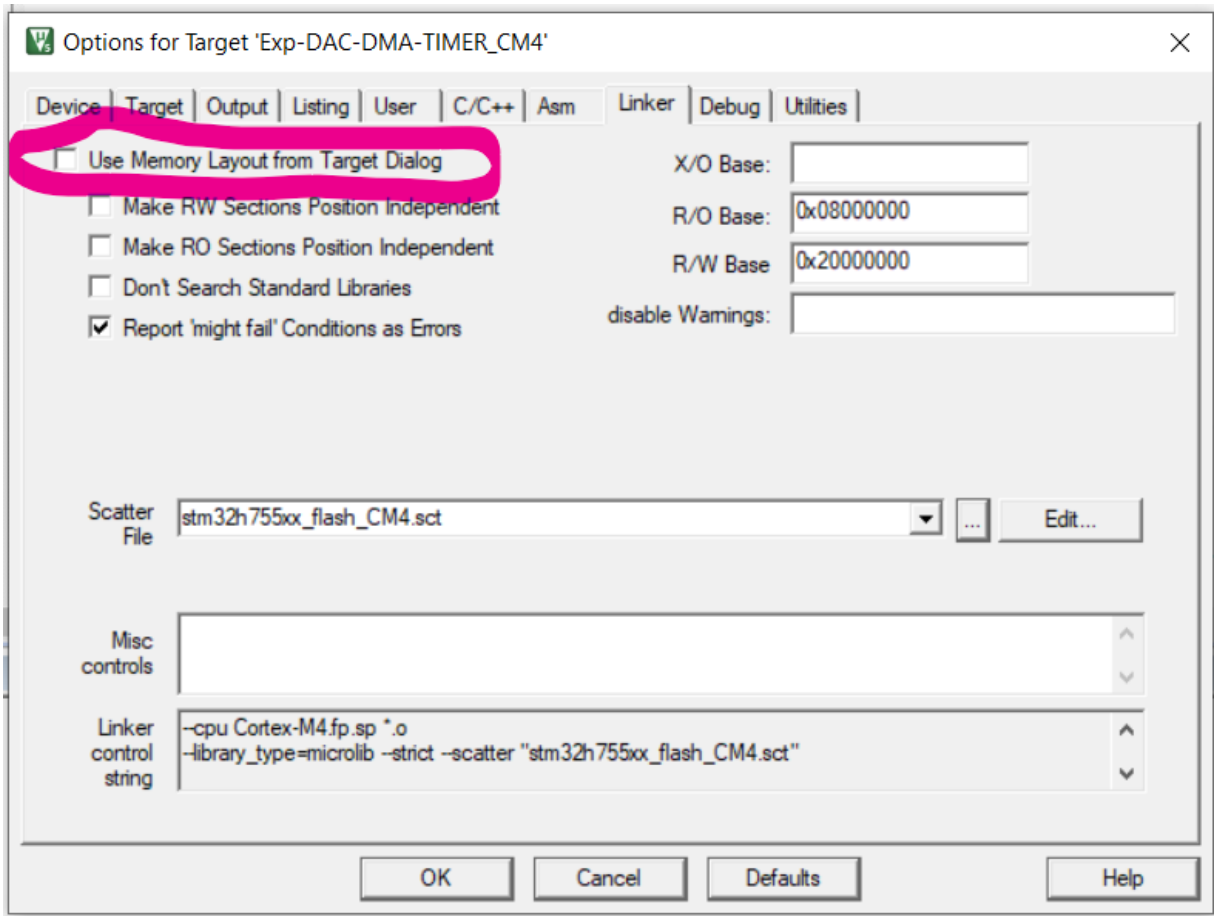


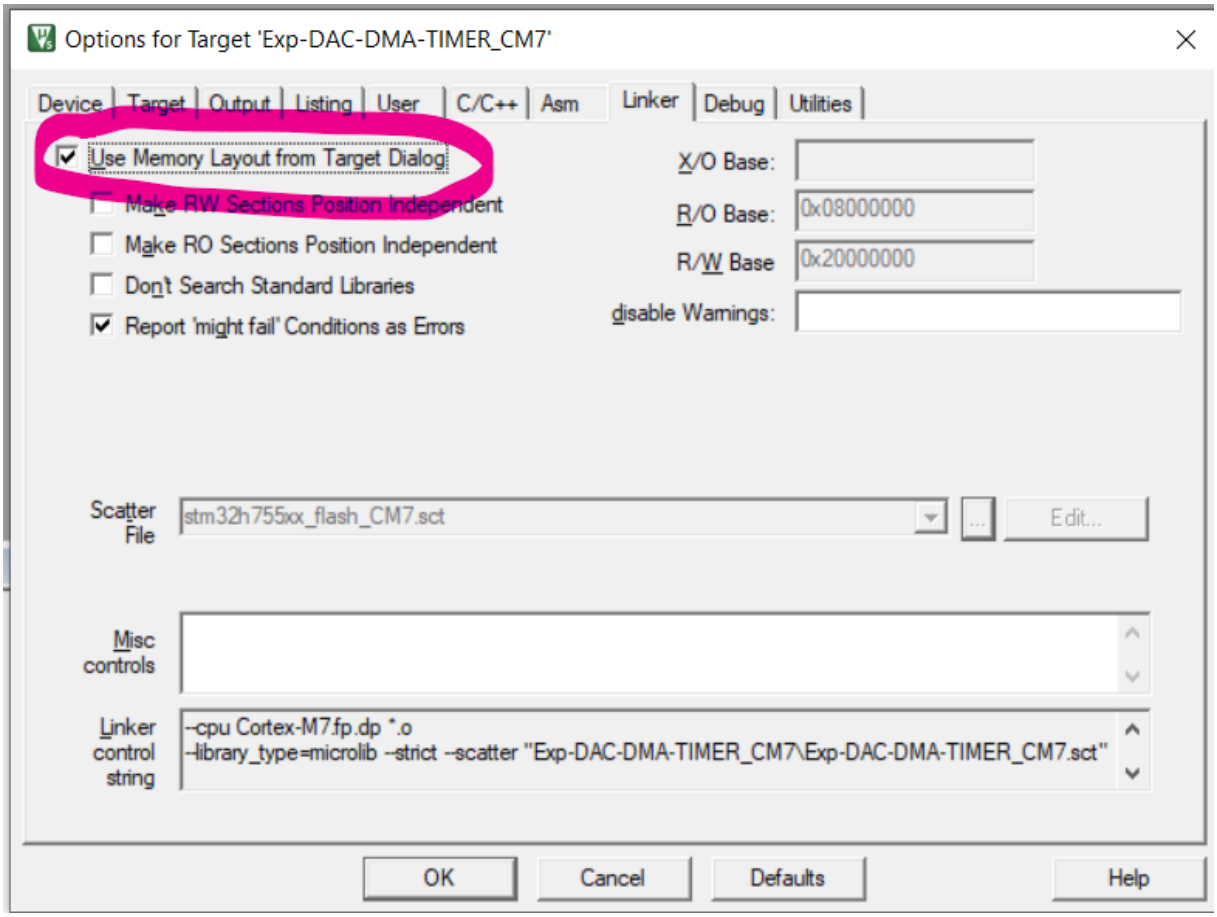
5) Mettre 0x10000000 dans start et 0x00001000 dans size et appuyer sur OK



6)
7) Appuyer encore sur OK dans la fenêtre mère.
8) Appuyer de nouveau sur F8 pour charger le programme du CM4

Si malgré ceci vous avez encore des messages d'erreur, revérifiez toutes les options comme indiqué au chapitre 3, opération 15) !!! Celle-ci est particulièrement délicate, pour CM4 il ne faut pas cocher la case, mais pour CM7 oui :


Options for Target 'Exp-DAC-DMA-TIMER_CM4'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Memory Layout from Target Dialog
Make RW Sections Position Independent
Make RO Sections Position Independent
Don't Search Standard Libraries
Report 'might fail' Conditions as Errors
X/O Base:
R/O Base:
0x08000000
R/W Base
0x20000000
disable Warnings:
Scatter File
stm32h755xx_flash_CM4.sct
...
Edit...
Misc controls
Linker control string
--cpu Cortex-M4.fp.sp *.o
--library_type=microlib --strict --scatter "stm32h755xx_flash_CM4.sct"
OK
Cancel
Defaults
Help


Options for Target 'Exp-DAC-DMA-TIMER_CM7'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Memory Layout from Target Dialog
Make RW Sections Position Independent
Make RO Sections Position Independent
Don't Search Standard Libraries
Report 'might fail' Conditions as Errors
X/O Base:
R/O Base:
0x08000000
R/W Base
0x20000000
disable Warnings:
Scatter File
stm32h755xx_flash_CM7.sct
...
Edit...
Misc controls
Linker control string
--cpu Cortex-M7.fp.dp *.o
--library_type=microlib --strict --scatter "Exp-DAC-DMA-TIMER_CM7\Exp-DAC-DMA-TIMER_CM7.sct"
OK
Cancel
Defaults
Help

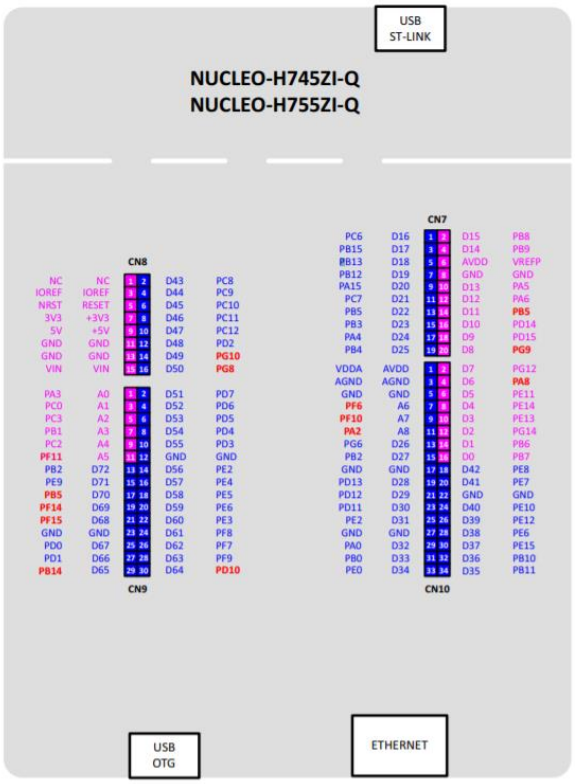
Enfin, vous n'êtes pas dispensé de regarder ce qui se passe dans le fichier .map que je vous laisse le soin de découvrir, il vous informera notamment du placement définitif du tableau sine_val dans le bon segment de mémoire 0x30000000 !

Pour ce premier test, Il n'y a rien du tout à écrire dans la boucle infinie, puisque c'est la fonction DMA qui se charge de tout envoyer au DAC !

Le processeur n'a donc aucun autre traitement à exécuter que la boucle infinie.

Arrivé à ce point du tutoriel, si vous branchez un oscilloscope (masse su CN8 position 11 et signal PA4 sur CN7 position 17) alors vous devriez observer une belle sinusoïde de 100 Hz.

NUCLEO-H745ZI-Q and NUCLEO-H755ZI-Q extension connectors



4.3 PETITE EVOLUTION

On voudrait bien voir évoluer un peu la fréquence de cette sinusoïde. Il y a plusieurs façon de le faire, en modifiant les paramètres du timer ou en redéfinissant le nombre d'échantillons de la courbe…

J'ai tellement eu de mal à faire fonctionner ce programme avec une tableau de variables, que c'est la deuxième solution que je vais choisir. Toutes les 3 secondes, je changerai le nombre d'échantillons pour faire passer la fréquence de 100→133→200→400→100 etc.

Rien de plus simple, il suffit que faire évoluer n cycliquement avec les valeurs suivantes : 100, 75, 50, 25, 100, 75… Ce qui donne le code suivant :

```
146   /* Infinite loop */
147   /* USER CODE BEGIN WHILE */
148   while (1)
149   {
150     /* USER CODE END WHILE */
151
152     /* USER CODE BEGIN 3 */
153     HAL_Delay(3000);
154     HAL_DAC_Stop_DMA(&hdac1,DAC1_CHANNEL_1);
155     if (n==25) n=125 ;
156     n = n-25;
157     get_sineval();
158     HAL_DAC_Start_DMA(&hdac1,DAC1_CHANNEL_1,sine_val,n,DAC_ALIGN_12B_R);
159   }
160   /* USER CODE END 3 */
```