جامعة حلب
كلية الهندسة الكهربائية والالكترونية
قسم هندسة التحكم والأتمتة
مخبر التحكم

مقرر المتحكمات المصغرة

الجلسة السادسة

السنة الرابعة ميكاترونيك

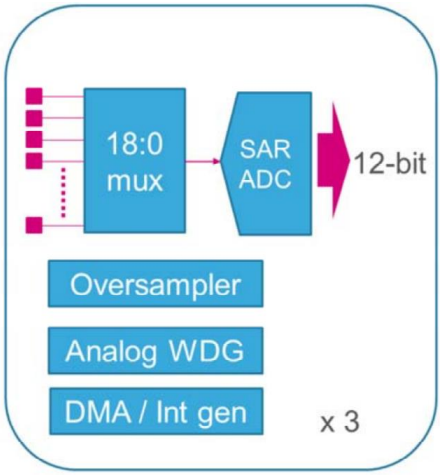
2023/2202

الغاية من الجلسة

1- التعرف على المبدلات التشابهية الرقمية في متحكمات STM32
2- أنماط عمليات التحويل ADC conversion modes
3- طرق قراءة المبدل التشابهي الرقمي
4- التطبيق العملي الأول: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWM من خلال مقاومة ضوئية موصولة على أحد أقطاب الدخل التشابهي باستخدام نمط الـــ Polling باستخدام البورد التطويري
5- التطبيق العملي الثاني: إعادة التطبيق السابق باستخدام نمط المقاطعة Interrupt
6- التطبيق العملي الثالث: مراقبة درجة حرارة الغرفة وعرضها على شاشة LCD

1- التعرف على المبدلات التشابهية الرقمية في متحكمات STM32:

المبدلات التشابهية الرقمية عبارة عن دارات الكترونية تقوم بتحويل الجهد التشابهي على دخلها إلى قيمة رقمية بالنظام الثنائي مقابلة لمستوى الجهد، فبمجرد قدح المبدل التشابهي الرقمي يبدأ بأخذ العينات samples ويقوم بعملية تدعى التكميم ليقابل كل مستوى من الجهد بما يناسبه من القيم الرقمية.تحتوي متحكمات STM32G0 على مبدل تشابهي رقمي وحيد من نوع Successive approximation ADC(SAR) بدقة 12بت وما يقارب الـ 19 قناة للمبدل



تسمح المبدلات التشابهية الرقمية للمتحكم STM32G0 باستقبال القيم التشابهية القادمة من الحساسات، حيث تقوم بتحويلها إلى القيم الرقيمة المقابلة لها، فللمبدل ما يقارب الـــ 19 قناة تحويل أي 19 دخل تشابهي للمتحكم بإمكانه استقبال القيم التشابهية من خلالها

يتم حساب جهد الدخل التشابهي من خلال العلاقة التالي:

$$Vin = ADC_{Res} * (V_{ref}/4096)$$

حيث :

ADC_{Res} : القيمة الرقمية الناتجة عن عملية التحويل التشابهي الرقمي

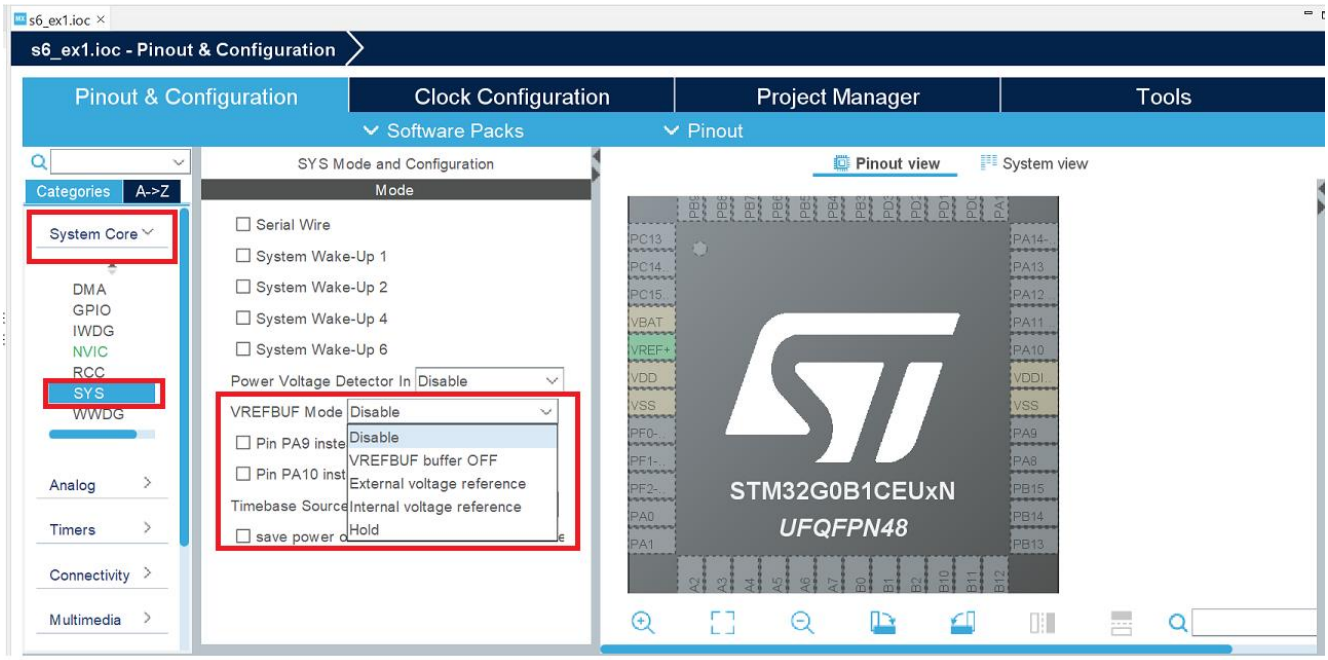
V_{ref} الجهد المرجعي

Features	Description
Input channel	Up to 16 external (GPIOs) and 3 internal channels
Type of conversion	12-bit successive approximation
Conversion time	400 ns, 2.5 Msamples/s (when f_{ADC_CLK} = 35 MHz, 12 bits)
Functional mode	Single, Continuous, Scan, and Discontinuous
Triggers	Software or external trigger (Timers & IOs)
Special functions	Analog watchdogs, Hardware oversampling, and Self-calibration
Data processing	Interrupt generation and DMA requests
Low-power modes	Wait, Auto-off, and Power-down

وهناك عدة احتمالات ممكنة للجهد المرجعي:

1. **Disabled** : في هذه الحالة يكون مولد الجهد المرجعي(VREF Buffer) في حالة فصل، ويكون قطب الجهد المرجعي +VREF متصل داخلياً مع VSSA
2. **External Voltage reference:** وهي الحالة **الافتراضية**، وفيها يكون مولد الجهد المرجعي(VREF Buffer) في حالة فصل، ويكون قطب الجهد المرجعي +VREF غير متصل مع أي تغذية، لذا **لابد من وصله مع جهد تغذية خارجي** على ألا تتجاوز قيمته الـ 3.3v
3. **Internal Voltage reference:** استخدام مولد الجهد المرجعي الداخلي: في هذه الحالة يكون مولد الجهد المرجعي(VREF Buffer) في حالة وصل، ويكون قطب الجهد المرجعي +VREF متصل داخلياً مع خرج هذا المولد، حيث يوجد داخل متحكمات STM32 مولد جهد مرجعي مدمج بداخلها يقوم بتوليد جهد مرجعي ثابت ومستقر حتى عند تغذيته من بطارية و يمكن استخدامه مع المبدل التشابهي الرقمي ADC والمبدل الرقمي التشابهي DAC ، و يعطي في خرجه إما v2.5 أو 2.048v ، كما يمكنه أن يغذي أحمال خارجية باستجرار تيار لا يتجاوز الـ mA4 .
4. **Hold Mode:** في هذه الحالة يكون مولد الجهد المرجعي(VREF Buffer) في حالة وصل، ويكون قطب الجهد المرجعي +VREF غير متصل مع أي تغذية، وفي هذه الحالة يجب وصل مكثفات على القطب +Vref

ENVR	HIZ	Configuration
0	0	VREF buffer OFF VREF+ pin pulled-down to VSSA
0	**1**	**External voltage reference mode (default):** • **VREF buffer OFF** • **VREF+ pin floating**
1	0	Internal voltage reference mode: • VREF buffer ON • VREF+ pin connected to the VREF buffer output
1	1	Hold mode: • VREF buffer ON • VREF+ pin floating. The voltage is held with an external capacitor



المعايرة الذاتية Self-calibration:

يوفر المبدل خاصـــية المعايرة الذاتية والتي تقلل بشـــكل كبير الأخطاء الناتجة عن تغيرات مكثف الشـــحن الداخلي (مكثف أخذ االعينات) internal capacitor، ونقوم عادة في بداية الكود باســـتخدام دالة من دوال HAL للقيام بعملية المعايرة:

```
HAL_ADCEx_Calibration_Start(&hadc1);
```

سرعة التحويل Conversion speed:

يحتـاج المبدل التشـــابهي الرقمي على الأقل 1.5clock cycles لأخذ العينـات و 12.5 clock cycles للتحويل من أجل دقة 12بت، بمعنى آخر ومن أجل تردد السـاعة الأعظمي للمبدل 35MHZ يمكن أن تصـل سرعة أخذ العينات إلى 2.5mega samples/s

طرق قراءة المبدل التشابهي الرقمي ADC conversion modes:

يوجد ثلاث طرق رئيسية لقراءة الـ ADC هي:

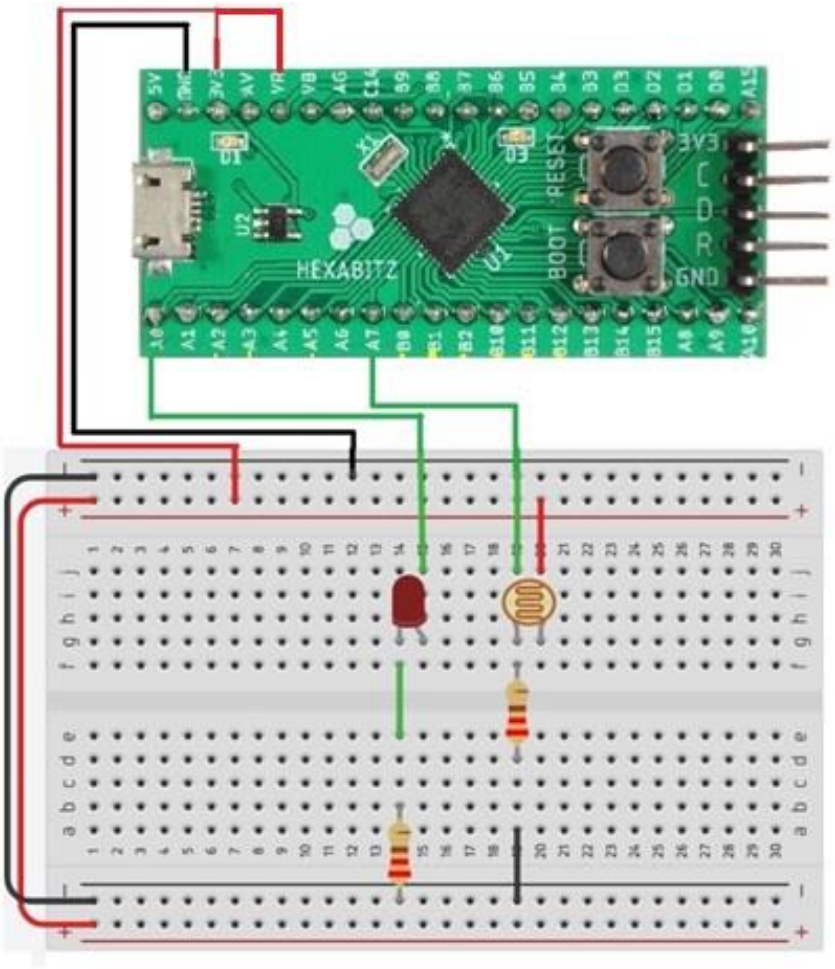
1. **Polling method:** تعتبر الطريقة الأسـهل في كتابة الكود لقراءة القيمة القادمة من إحدى القنوات التشابهية، ولكنها ليست الأكثر فعالية ، حيث علينا أن نبدأ بعملية التحويل وتتوقف الـ CPU عن تنفيذ الكود وتنتظر لحين الانتهاء من عملية التحويل حينها يمكن للـ CPU استكمال تنفيذ الكود الرئيسي.
2. **The interrupt Method:** تعتبر هذه الطريقة طريقة فعالة لاستخدام المبدل التشابهي الرقمي ، حيث نقوم بقدح المبدل فقط و يمكن للـــ CPU أن تسـتكمل تنفيذ الكود والمهام المطلوبة منها لحين انتهاء عملية التحويل عندها سيقوم المبدل بطلب مقاطعة وستتوجه الـ CPU لبرنامج خدمة المقاطعة وتحتفظ بناتج عملية التحويل ليتم معالجته.
3. **DMA Method**

2- **التطبيق العملي الأول:** سـنقوم في هذا التطبيق بالتحكم بشـدة إضـاءة ليد موصـول على أحد أقطاب الـ PWM (القطب PA0) من خلال مقاومة ضوئية موصولة على أحد أقطاب الدخل التشابهي
سنقوم بتنفيذ المشروع وفقاً للتسلسل التالي:

- ضبط تردد ساعة المتحكم
- ضـبط قطب الدخل التشـابهي (PA7) في نمط التحويل لمرة واحدة Single conversion mode حيث سنربط مقاومة ضوئية معه.
- ضبط الـ timer2 في نمط PWM على القناة CH1 (وسنربط معه ليد).

سـنقوم بتنفيذ المشـروع بطريقتي Polling ,interrupt، حيث سـنقوم في البداية بقراءة القيمة القادمة من المقاومة المتغيرة الموجودة على القناة CH7 للمبدل التشـابهي ومن ثم سـنقوم بإسـناد هذه القيمة لمسـجل المؤقت CCR والذي من خلاله يتم تحديد دورة التشغيل dutycycle والتي تحدد شدة إضاءة الليد.

الطريقة الأولى: استخدام نمط الـ Polling:



سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

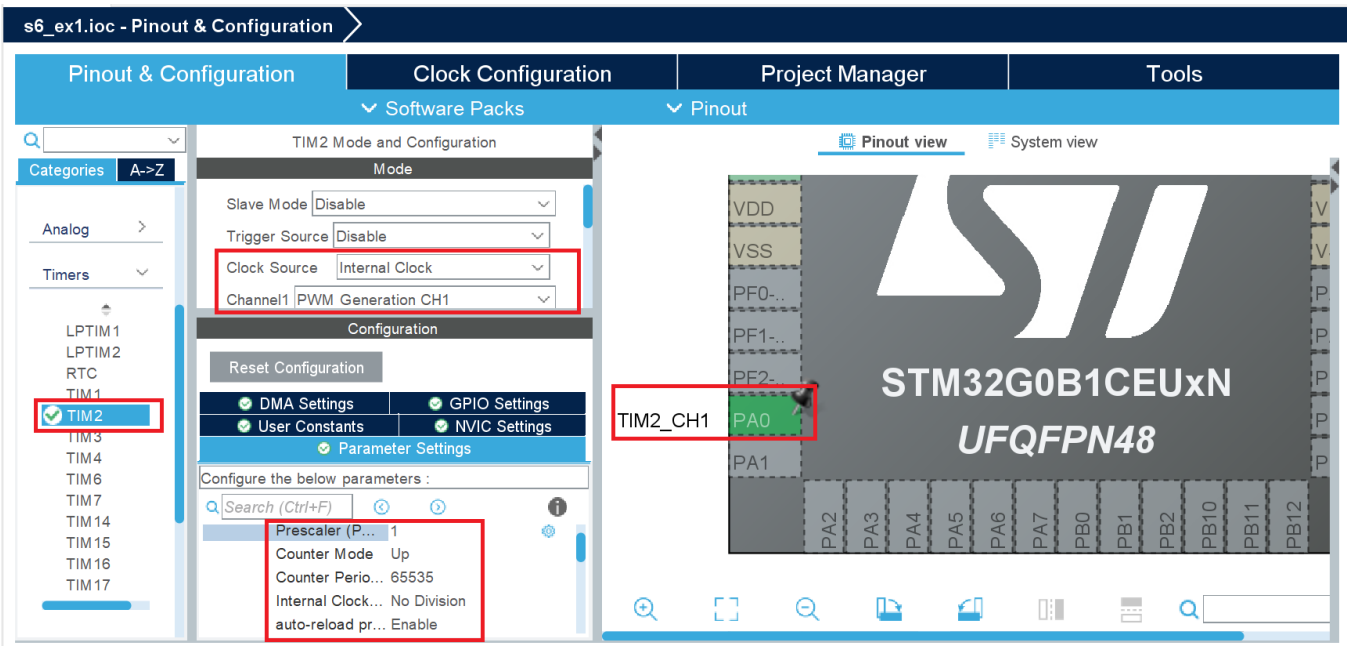
الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشـــاء مشـــروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر أو من خلال اختيار اسم اللوحة المستخدمة وهو في حالتنا STM32G0B1CEU6N .

الخطوة الثانية: ضبط إعدادات المؤقت ليعمل في نمط PWM

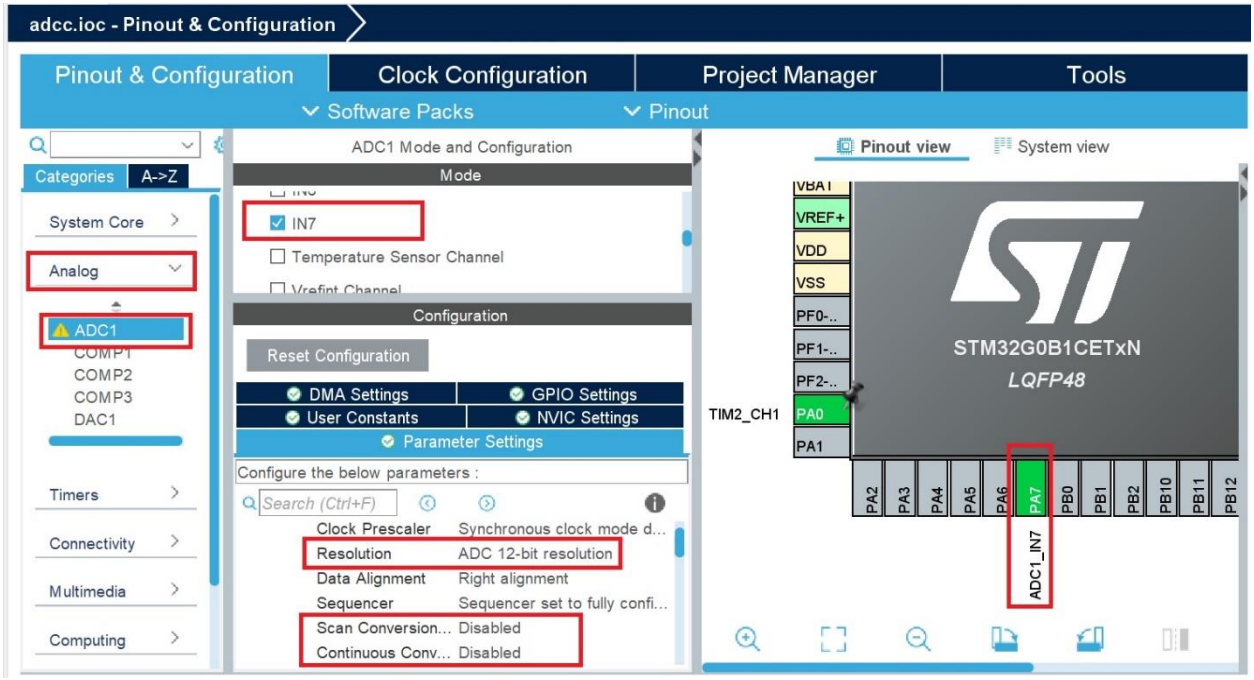
نقوم بضبط مصدر الساعة للمؤقت على الساعة الداخلية للنظام internal clock ، نقوم بتفعيل القناة CH1 لتكون القناة التي سيتم إخراج إشارة الـــ PWM عليها، نضبط القيمة العظمى للمسجل ARR على القيمة 65535 ، نفعل خاصـية Auto Reload preload ونختار نمط إشـارة الـــــ PWM وبفرض

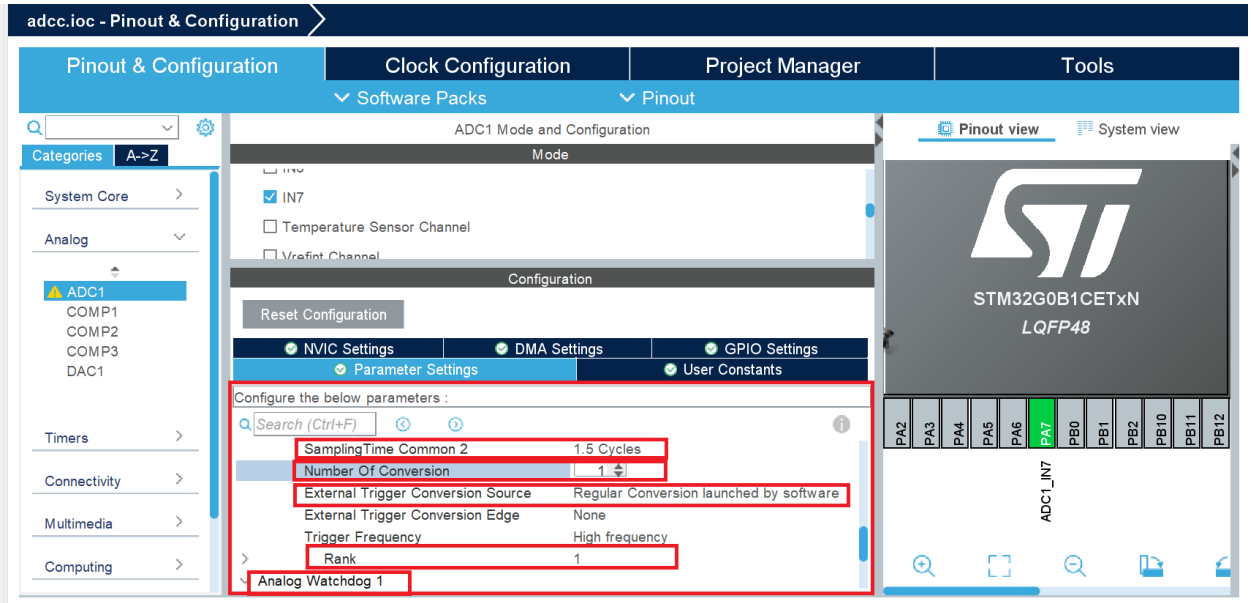
أن F_{CLK} =64MHZ ، Prescaler=1، ARR=65535، يصبح تردد نبضات الـ PWM:

$$F_{PWM} = \frac{64 \times (10\text{^}6)}{(65535 + 1) \times (1 + 1)} = 488.28HZ$$

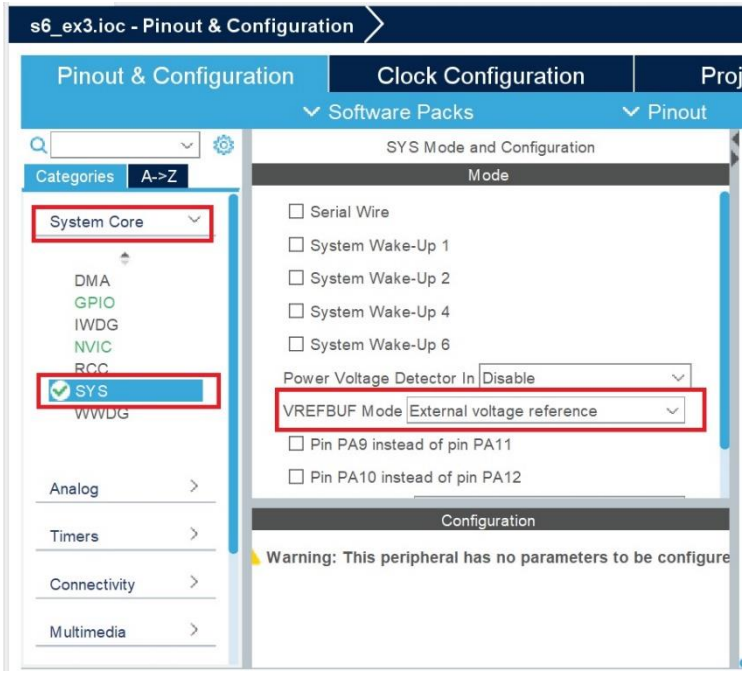


الخطوة الثالثة: قم بضبط إعدادات المبدل التشابهي الرقمي ADC على القناة التشابهية السابعة CH7 بالشكل التالي:

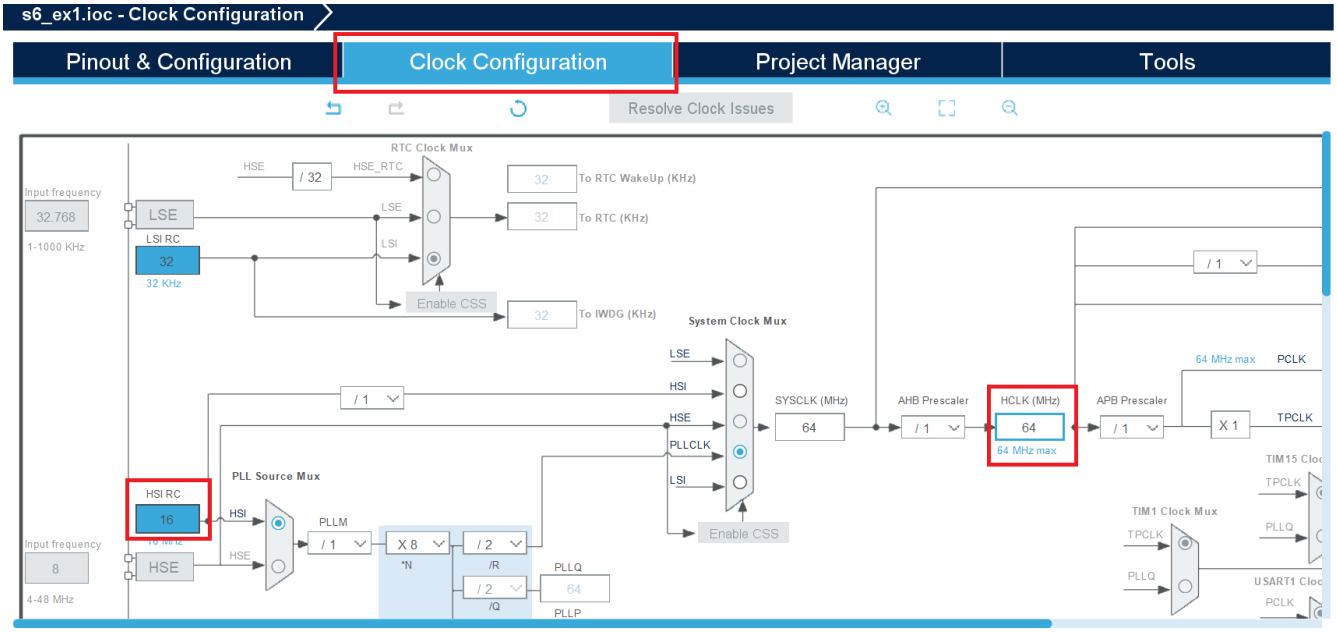




الخطوة الرابعة: قم بضبط إعدادات الجهد المرجعي



الخطوة الخامسة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

يصبح الكود النهائي بالشكل التالي:

تعريف المبدل التشابهي ADC1
تعريف المؤقت TIM2
	`#include "main.h"`
	
	`ADC_HandleTypeDef hadc1;`
	`TIM_HandleTypeDef htim2;`

التصريح عن الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	`void SystemClock_Config(void);`
	`static void MX_GPIO_Init(void);`
	`static void MX_ADC1_Init(void);`
	`static void MX_TIM2_Init(void);`

التصريح عن متحول AD_RES لنخزن فيه لاحقاً القيمة الناتجة عن المحول التشابهي الرقمي	`int main(void)`
	`{`
	`  uint16_t AD_RES = 0;`

استدعاء الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	`  HAL_Init();`
	`  SystemClock_Config();`
	`  MX_GPIO_Init();`
	`  MX_ADC1_Init();`
	`  MX_TIM2_Init();`

بدء عمل المؤقت في نمط الـ PWM وعلى القناة الأولى	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1); // Calibrate The ADC On Power-Up For Better Accuracy
بدء المعايرة الذاتية للمبدل التشابهي الرقمي adc1	HAL_ADCEx_Calibration_Start(&hadc1);

إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل	while (1)
	{
	// Start ADC Conversion
	HAL_ADC_Start(&hadc1);

طلب عملية تحويل من المبدل التشابهي الرقمي	// Poll ADC1 Perihperal & TimeOut = 1mSec
	HAL_ADC_PollForConversion(&hadc1, 1);

إسناد ناتج عملية التحويل إلى المتغير AD_RES	// Read The ADC Conversion Result & Map It To PWM DutyCycle
	AD_RES = HAL_ADC_GetValue(&hadc1);

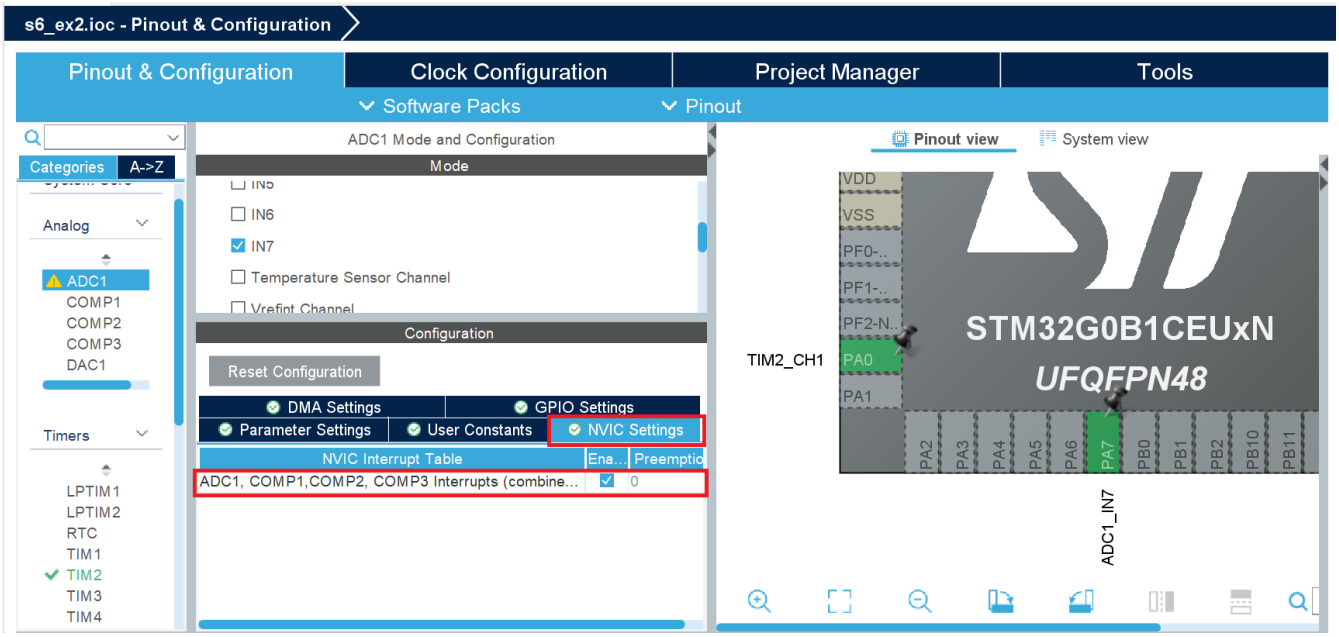
إسناد قيمة المتغير AD_RES التي تعبر عن قيمة المقاومة الضوئية ، إلى المسجل CCR1 من المؤقت TIM2 ، المسؤول عن دورة التشغيل dutycycle	TIM2->CCR1 = (AD_RES<<4);
إجراء تأخير زمني بمقدار 1msec	HAL_Delay(1); } }

التطبيق العملي الثاني: إعادة التطبيق الأول ولكن باستخدام نمط الـ Interrupt:

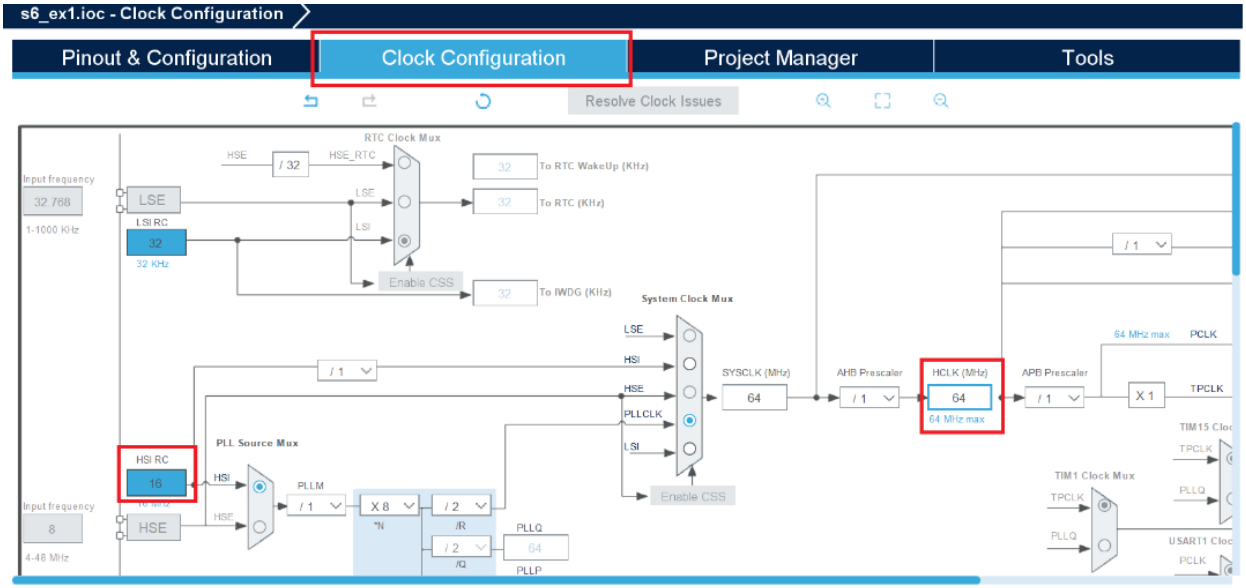
سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

الخطوة الأولى والثانية هي نفس الخطوات الموجودة في الطريقة الأولى

الخطوة الثالثة: ضبط إعدادات المبدل التشابهي بنفس الإعدادات السابقة ، فقط سنفعل المقاطعة الخاصة بالمبدل



الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

يصبح الكود النهائي بالشكل التالي:

```
#include "main.h"

uint16_t AD_RES = 0;
```

التصريح عن متحول عام (لأننا سنستخدمه في الدالة الرئيسية وفي برنامج خدمة المقاطعة) AD_RES لنخزن فيه لاحقاً القيمة الناتجة عن المحول التشابهي الرقمي	ADC_HandleTypeDef hadc1; TIM_HandleTypeDef htim2;
تعريف المبدل التشابهي ADC1 تعريف المؤقت TIM2	**void SystemClock_Config(void);** **static void MX_GPIO_Init(void);** **static void MX_ADC1_Init(void);** **static void MX_TIM2_Init(void);**
التصريح عن الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	**int main(void)** {

تهيئة مكتبة HAL
استدعاء الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2
	HAL_Init();
	SystemClock_Config();
	MX_GPIO_Init();
	MX_ADC1_Init();
	MX_TIM2_Init();
	
	HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
	// Calibrate The ADC On Power-Up For Better Accuracy
	HAL_ADCEx_Calibration_Start(&hadc1);

بدء عمل المؤقت في نمط الـ PWM وعلى القناة الأولى
 	
بدء المعايرة الذاتية للمبدل التشابهي الرقمي adc1
	while (1)
	{
	// Start ADC Conversion
	HAL_ADC_Start_IT(&hadc1);
	// Update The PWM Duty Cycle With Latest ADC Conversion Result

إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل	TIM2->CCR1 = (AD_RES<<4); HAL_Delay(1); } }

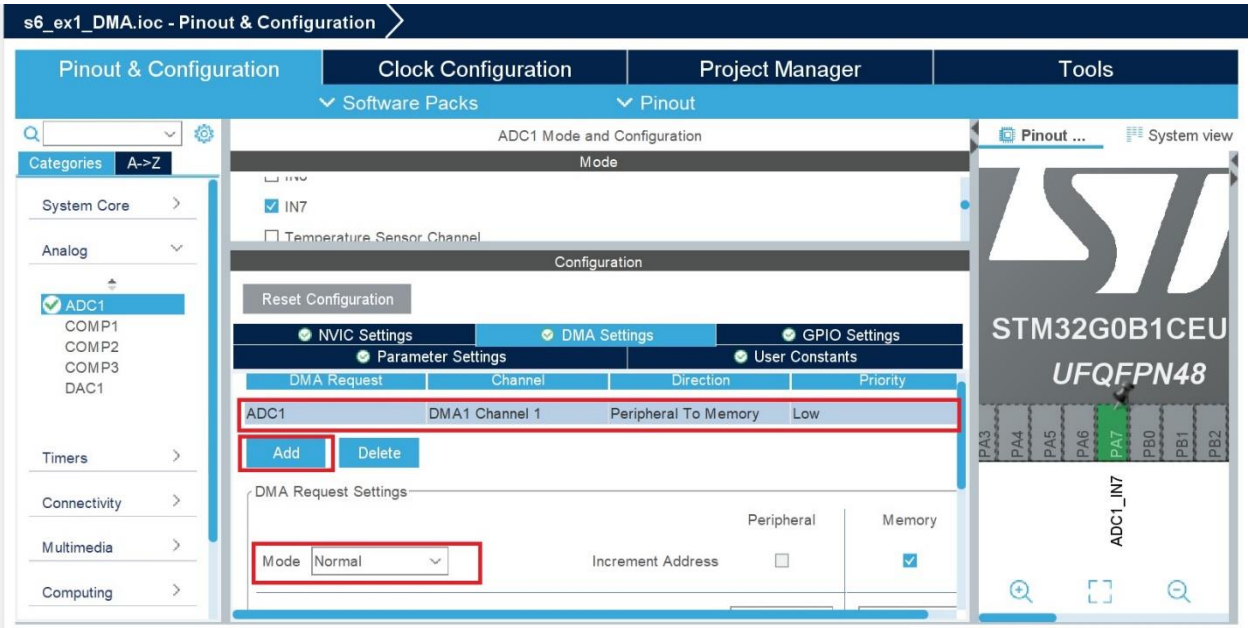
إسناد قيمة المتغير AD_RES التي تعبر عن قيمة المقاومة الضوئية ، إلى المسجل CCR1 من المؤقت TIM2 ، المسؤول عن دورة التشغيل dutycycle
إجراء تأخير زمني بمقدار 1sec
 	
برنامج خدمة المقاطعة والذي يتم استدعاؤه عند إتمام عملية التحويل التشابهي الرقمي
 	
إسناد ناتج عملية التحويل إلى المتغير AD_RES (نحصل على ناتج التحويل التشابهي الرقمي من برنامج خدمة المقاطعة)
	void
	HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
	{
	
	// Read & Update The ADC Result
	AD_RES = HAL_ADC_GetValue(&hadc1);
	}

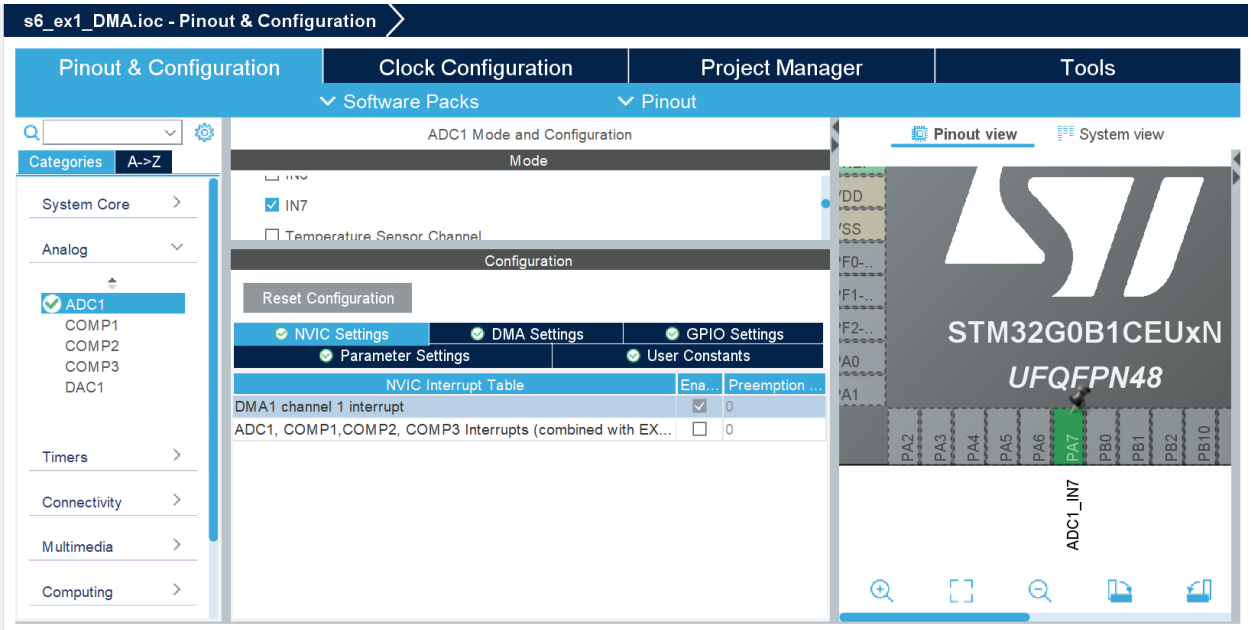
التطبيق العملي الثالث: إعادة التطبيق الأول ولكن باستخدام نمط الـ DMA:

سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

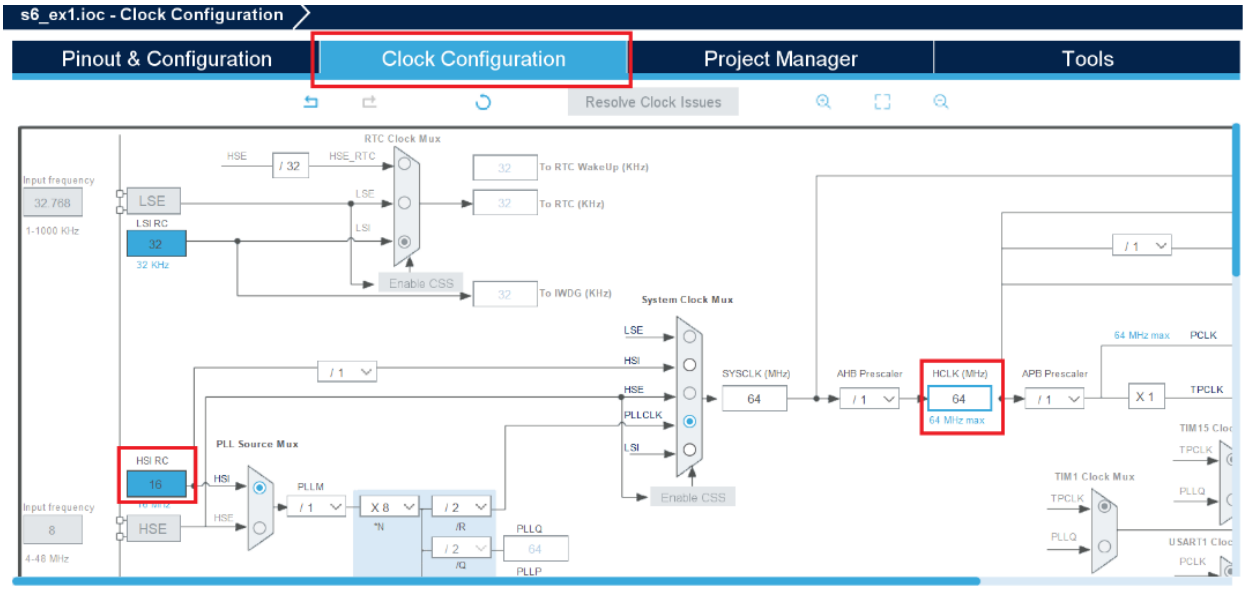
الخطوة الأولى والثانية هي نفس الخطوات الموجودة في الطريقة الأولى

الخطوة الثالثة: ضبط إعدادات المبدل التشابهي بنفس الإعدادات السابقة ، فقط سنفعل نمط الـ DMA من خلال إضافة قناة إلى DMA ونلاحظ ان مقاطعة الـ DMA تتفعل بشكل تلقائي





الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

يصبح الكود النهائي بالشكل التالي:

التصريح عن متحول عام (لأننا سنستخدمه في الدالة الرئيسية وفي برنامج خدمة المقاطعة) AD_RES لنخزن فيه لاحقاً القيمة الناتجة عن المحول التشابهي الرقمي
تعريف المبدل التشابهي ADC1
تعريف قناة الـ DMA المستخدمة مع المبدل
تعريف المؤقت TIM2
	`#include "main.h"`
	
	`uint16_t AD_RES = 0;`
	
	`ADC_HandleTypeDef hadc1;`
	
	`DMA_HandleTypeDef hdma_adc1;`
	
	`TIM_HandleTypeDef htim2;`

التصريح عن الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	`void SystemClock_Config(void);`
	`static void MX_GPIO_Init(void);`
	`static void MX_DMA_Init(void);`
	`static void MX_ADC1_Init(void);`
	`static void MX_TIM2_Init(void);`

تهيئة مكتبة HAL
استدعاء الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2
	`int main(void)`
	`{`
	`  HAL_Init();`
	`  SystemClock_Config();`
	`  MX_GPIO_Init();`
	`  MX_DMA_Init();`
	`  MX_ADC1_Init();`
	`  MX_TIM2_Init();`

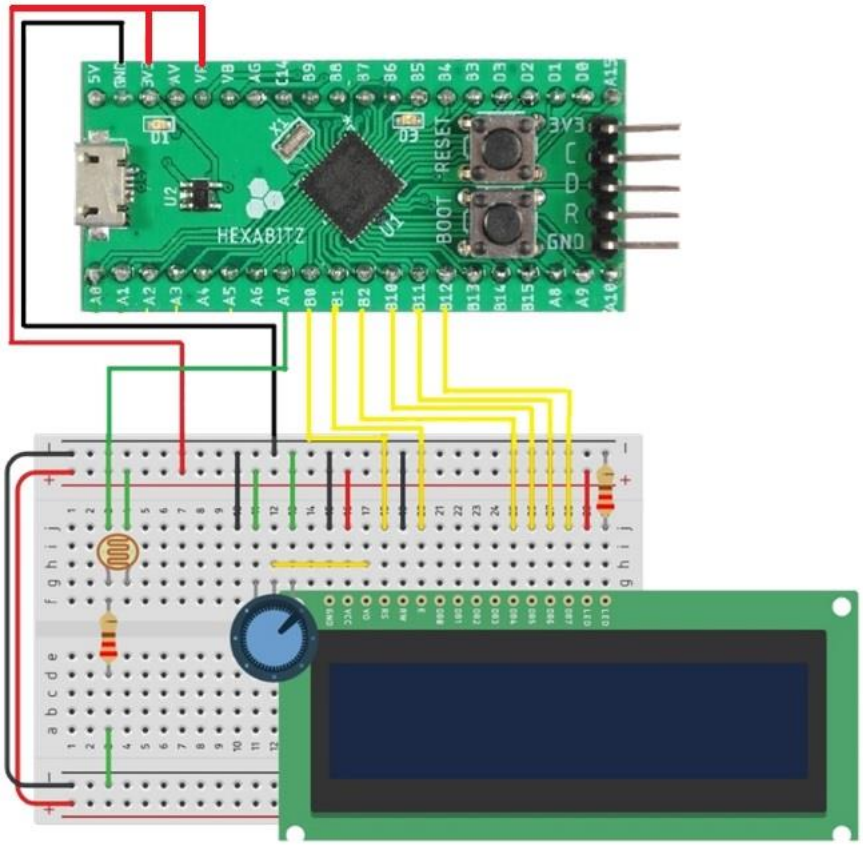
بدء عمل المؤقت في نمط الـ PWM وعلى القناة الأولى	`  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);` `  // Calibrate The ADC On Power-Up For Better Accuracy`
بدء المعايرة الذاتية للمبدل التشابهي الرقمي adc1	`  HAL_ADCEx_Calibration_Start(&hadc1);` `  while (1)`

إعطاء إشارة البدء للمبدل التشابهي الرقمي حيث سيتم تخزين ناتج التحويل تلقائياً ضمن المتغير AD_RES	`  {`
	`  // Start ADC Conversion`
	`  // Pass (The ADC Instance, Result Buffer Address, Buffer Length)`
	`  HAL_ADC_Start_DMA(&hadc1, &AD_RES, 1);`
	`  HAL_Delay(1);`
	`  }`
	`}`

برنامج خدمة المقاطعة والذي يتم استدعاؤه عند إتمام عملية التحويل التشابهي الرقمي	`void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)` `{`

إسناد ناتج عملية التحويل من المتغير AD_RES إلى المسجل CCR1 ليقوم بتعديل الـ duty_cycle تبعاً لها	`  // Conversion Complete & DMA Transfer Complete As Well`
	`  // So The AD_RES Is Now Updated & Let's Move IT To The PWM CCR1`
	`  // Update The PWM Duty Cycle With Latest ADC Conversion Result`
	`  TIM2->CCR1 = (AD_RES<<4);`
	`}`

3- **التطبيق العملي الرابع:** ســـنقوم في هذا التطبيق بمراقبة جهد متغير موصـــول على أحد أقطاب الدخل التشابهي ADC وعرض القيمة على شاشة lcd.



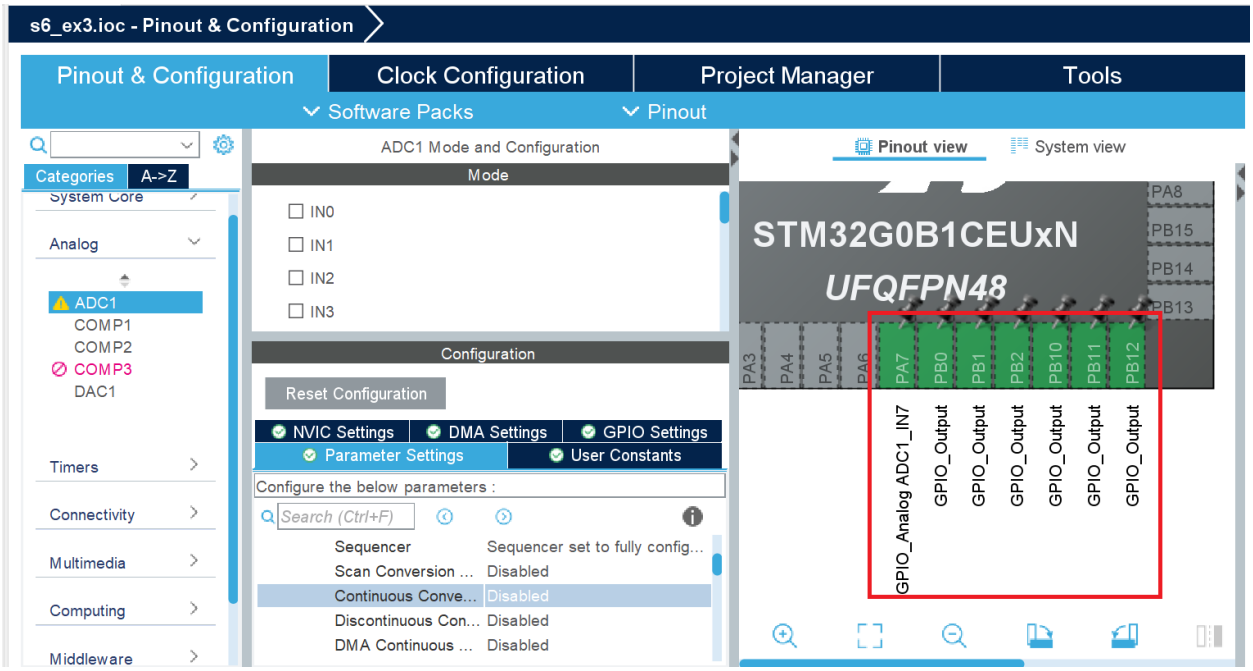
سنقوم بتنفيذ المشروع وفقاً للتسلسل التالي:

- ضبط تردد ساعة المتحكم
- ضبط قطب الدخل التشابهي (PA7) في نمط polling .
- إضافة مكتبة الـ lcd إلى الكود

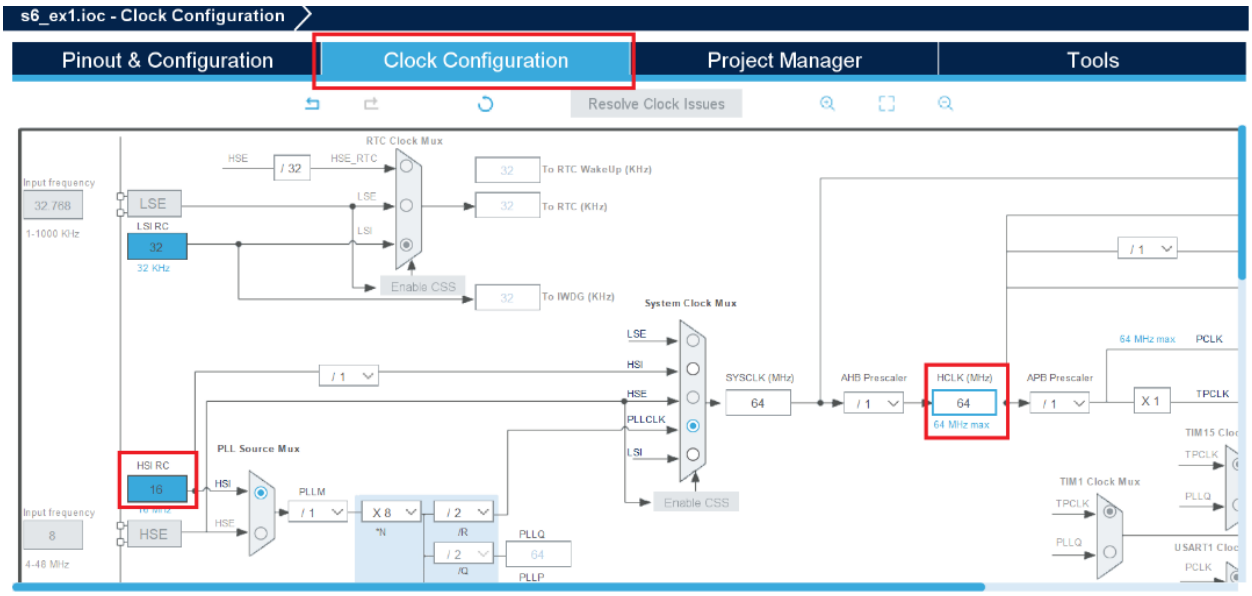
ضبط إعدادات المشروع:

الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشـــاء مشـــروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر أو من خلال اختيار اسم اللوحة المستخدمة وهي في حالتنا STM32G0B1CEU6N كما في الشكل التالي:

الخطوة الثالثة: قم بضبط إعدادات المبدل التشابهي الرقمي ADC على القناة التشابهية السابعة CH7 ، اختر الأقطاب PB10:PB12، PB0:PB2 لضبطها كأقطاب خرج ليتم وصل شاشة الـ lcd معها، بالشكل التالي:



الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s، ثم إضافة مكتبة الـ lcd والتعليمات الخاصة ببدء المبدل التشابهي ومعايرته و أخذ القراءات من المقاومة المتغيرة وعرضها على شاشة الـ lcd

```
#include "main.h"
#include "lcd_txt.h"
```

```
#include<stdio.h>
ADC_HandleTypeDef hadc1;
uint16_t  adc_value= 0;
uint16_t voltage=0;
int8_t* str;
int8_t *str1= (int8_t*)"voltage=";
int8_t *str2= (int8_t*)"mv";
char buffer[16];
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_DMA_Init(void);
int main(void)
{
        HAL_Init();
        SystemClock_Config();
        MX_GPIO_Init();
        MX_ADC1_Init();
        lcd_init();
        lcd_clear();
 // Calibrate The ADC On Power-Up For Better Accuracy
        HAL_ADCEx_Calibration_Start(&hadc1);
        while (1)
        {
           // Start ADC Conversion
            HAL_ADC_Start(&hadc1);
           // Poll ADC1 Perihperal & TimeOut = 1mSec
            HAL_ADC_PollForConversion(&hadc1, 1);
           // Read The ADC Conversion Result
          adc_value = HAL_ADC_GetValue(&hadc1);
          voltage= adc_value*(0.8057);

          snprintf(buffer, 10, "%d", voltage);
          str=(int8_t*)buffer;
          lcd_puts(0,0,str1);
          lcd_puts(0,9,str);
          lcd_puts(0,14,str2);
            HAL_Delay(1);
        }
}
```

الخطوة السادسة: تعديل أقطاب الـ lcd بما يناسب التوصيل العملي للشاشة مع المتحكم

lcd_txt.h

```
/*Note: Comment which not use */
#define LCD16xN //For lcd16x2 or lcd16x4
//#define LCD20xN //For lcd20x4
/*------------- Define For Connection -----------------*/
#define RS_PORT     GPIOB
#define RS_PIN      GPIO_PIN_0
#define EN_PORT     GPIOB
#define EN_PIN      GPIO_PIN_1
#define D7_PORT     GPIOB
#define D7_PIN      GPIO_PIN_12
#define D6_PORT     GPIOB
#define D6_PIN      GPIO_PIN_11
#define D5_PORT     GPIOB
#define D5_PIN      GPIO_PIN_10
#define D4_PORT     GPIOB
#define D4_PIN      GPIO PIN 2
```

4- التطبيق العملي الخامس:أعد التطبيق السابق باستخدام المقاطعة

```
#include "main.h"
#include "lcd_txt.h"
#include<stdio.h>
ADC_HandleTypeDef hadc1;
uint16_t  adc_value= 0;
uint16_t voltage=0;
int8_t* str;
int8_t *str1= (int8_t*)"voltage=";
int8_t *str2= (int8_t*)"mv";
char buffer[16];
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    // Read & Update The ADC Result
    adc_value = HAL_ADC_GetValue(&hadc1);
}
int main(void)
{
        HAL_Init();
        SystemClock_Config();
```

```
        MX_GPIO_Init();
        MX_ADC1_Init();
        lcd_init();
        lcd_clear();
 // Calibrate The ADC On Power-Up For Better Accuracy
        HAL_ADCEx_Calibration_Start(&hadc1);
        while (1)
        {
           // Start ADC Conversion
         HAL_ADC_Start_IT(&hadc1);
          // Read The ADC Conversion Result
            voltage= adc_value*(0.8057);

          snprintf(buffer, 10, "%d", voltage);
          str=(int8_t*)buffer;
          lcd_puts(0,0,str1);
          lcd_puts(0,9,str);
          lcd_puts(0,14,str2);
            HAL_Delay(1);     }}
```

5- التطبيق العملي السادس:أعد التطبيق السابق باستخدام نمط الـ DMA

```
#include "main.h"
#include "lcd_txt.h"
#include<stdio.h>


ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

uint16_t  adc_value= 0;
uint16_t voltage=0;
int8_t* str;
int8_t *str1= (int8_t*)"voltage=";
int8_t *str2= (int8_t*)"mv";
char buffer[16];

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
```

```
static void MX_ADC1_Init(void);

int main(void)
{
   HAL_Init();
 SystemClock_Config();
 MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  lcd_init();
  lcd_clear();
  HAL_ADCEx_Calibration_Start(&hadc1);
  while (1)
  {
       // Start ADC Conversion
                        HAL_ADC_Start_DMA(&hadc1, & adc_value, 1);
                  // Read The ADC Conversion Result
                     voltage= adc_value*(0.8057);

                  snprintf(buffer, 10, "%d", voltage);
                  str=(int8_t*)buffer;
                   lcd_puts(0,0,str1);
                   lcd_puts(0,9,str);
                   lcd_puts(0,14,str2);
                    HAL_Delay(100);

  }
  /* USER CODE END 3 */
                                                                    }
```