

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

## مقرر المتحكمات المصغرة الجلسة السابعة

السنة الرابعة ميكاترونيك

**2023/2202**

## Universal Asynchronous Serial Communications

### USART/UART

#### محتويات الجلسة:

- 1- مقدمة
  - 2- مفهوم UART و USART
  - 3- أنماط العمل المختلفة للـ UART في متحكمات STM32
  - 4- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ Polling
  - 5- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt
  - 6- التطبيق العملي الأول: مراقبة عمل أي كود (Debugging) من خلال نافذة الاتصال التسلسلي
  - 7- التطبيق العملي الثاني: إعادة التطبيق السابق باستخدام نمط الـ interrupt
- الأدوات اللازمة للجلسة:

- البورد التطويري من شركة Hexabit
- كبل Type-A to Mini-B
- ليدات
- كبل FTDI للاتصال التسلسلي بالبورد

#### 1- مقدمة :

في هذه الأيام يوجد العديد من بروتوكولات الاتصال التسلسلية Serial communication Protocols ، حيث يركز أغلبها على سرعة نقل البيانات مثل USB 3.0/ USB 2.0 وغيرها...، إحدى طرق الاتصال التي تم إنشاؤها قديماً وما زالت تستخدم حتى الآن للربط بين المتحكمات المصغرة هي Universal Synchronous/Asynchronous Receiver/Transmitter Interface (USART).

Synchronous: هو الإرسال والاستقبال المتزامن المبني على وجود clock بين المرسل والمستقبل.

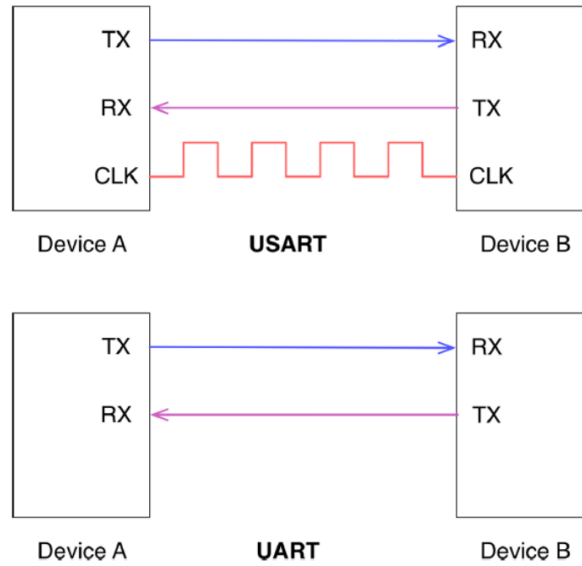
Asynchronous: لا يعتمد على clock وإنما يتم الاكتفاء بإرسال البيانات على خط الإرسال ويتم استقبالها على خط الاستقبال.

كل متحكم STM32 يحتوي على الأقل على وحدة طرفية UART واحدة، وأغلب متحكمات STM32 توفر على الأقل اثنتين من UART/USART ، وأخرى توفر لحد 8 وحدات طرفية UART/USART، حيث توفر لوحتنا التطويرية التي تحتوي على متحكم STM32G0B1CEU6N على 6 وحدات طرفية للـ UART/USART.

#### 2- مفهوم UART و USART:

- عندما تريد نقل البيانات ما بين جهازين أو أكثر، يوجد طريقتين لإرسال البيانات الأولى وهي: نقل البيانات على التوازي Parallel : في هذه الطريقة توجد مجموعة من خطوط البيانات على حسب طول البيانات التي سيتم إرسالها (مثلاً 8 خطوط بيانات في حال إرسال بيانات بطول 8 بت).

- نقل البيانات على التسلسل Serial : وفي هذه الطريقة يتم إرسال البيانات على التوالي بت تلو الآخر باستخدام خط بيانات واحد حيث يكون أحد الجهازين المتصلين مرسل والآخر مستقبل.
- في وضع Synchronous يتم مشاركة clock ما بين المرسل والمستقبل والتي يتم إنشاؤها دائماً باستخدام الجهاز الذي يدير الاتصال، أما في حالة الإرسال الغير متزامن (Asynchronous) يتم الاستغناء عن الـ clock حيث يتم استخدام بت عند بداية الإرسال Start Bit و بت عن انتهاء الإرسال Stop Bit.



Pin number								Pin name (function upon reset)	Pin type	I/O structure	Note	Alternate functions	Additional functions
WLCSP25	UFQFPN28 - GP	UFQFPN28 - N	UFQFPN32 - GP	UFQFPN32 - N	LQFP48 / UFQFPN48	UFPGA64	LQFP64						
D4	7	7	8	8	12	H3	18	PA1	I/O	FT_a	-	SPI1_SCK/I2S1_CK, USART2_RTS_DE_CK, TIM2_CH2, USART4_RX, TIM15_CH1N, I2C1_SMBA, EVENTOUT	COMP1_INP, ADC_IN1
E4	8	8	9	9	13	G3	19	<u>PA2</u>	I/O	FT_a	-	SPI1_MOSI/I2S1_SD, <u>USART2_TX</u> , TIM2_CH3, UCPD1_FRSTX, TIM15_CH1, LPUART1_TX, COMP2_OUT	COMP2_INM, ADC_IN2, WKUP4, LSCO
C3	9	9	10	10	14	F3	20	<u>PA3</u>	I/O	FT_a	-	SPI2_MISO, <u>USART2_RX</u> , TIM2_CH4, UCPD2_FRSTX, TIM15_CH2, LPUART1_RX, EVENTOUT	COMP2_INP, ADC_IN3
-	-	-	-	-	15	H4	21	PA4	I/O	TT_a	-	SPI1_NSS/I2S1_WS, SPI2_MOSI, TIM14_CH1, LPTIM2_OUT, UCPD2_FRSTX, EVENTOUT	ADC_IN4, DAC_OUT1, RTC_OUT2

### 3- أنماط العمل المختلفة للـ UART في متحكمات STM32:

توجد 3 أنماط عمل مختلفة للـ UART وهي:

#### 1. نمط Polling Mode:

يسمى أيضاً Blocking Mode ، في هذا النمط يتم تفحص عملية إرسال واستقبال البيانات بشكل مستمر ، حيث ينتظر المعالج لحين انتهاء عملية الإرسال مما يؤدي إلى تأخير معالجة باقي التعليمات وتنفيذ المهام ، وهو نمط العمل الأبسط من ناحية الكود ومن ناحية الـ Hardware — ويستخدّم عندما تكون كمية البيانات المتبادلة ليست كبيرة نسبياً ولا تمثل أهمية عالية من ناحية المعالجة.

#### 2. نمط المقاطعة Interrupt Mode:

ويسمى أيضاً non-Blocking Mode ، في هذا النمط لا يتم الانتظار وتفقد البيانات من حين لآخر للتأكد من عملية الإرسال والاستقبال، حيث عند الانتهاء من إرسال البيانات يتم تفعيل مقاطعة تفيد بانتهاء عملية الإرسال ، وهذا النمط من العمل أفضل من ناحية المعالجة ملائم عندما يكون معدل نقل البيانات صغير نسبياً (أقل من 38400Bps).

#### 3. نمط DMA:

وهو النمط الأفضل من ناحية إنتاجية نقل البيانات ومن ناحية سرعة نقل البيانات وعندما نريد تحرير المتحكم من الحمل الإضافي الذي ينتج عن من إحضار البيانات من RAM ومعالجتها، فالـ DMA يقوم بالوصول إلى الذاكرة RAM بدون احتياج أي جهد من المعالج لعمل ذلك، وبدون نمط الـ DMA لا يمكن التعامل مع السرعات العالية في الـ UART. سنشرح بالتفصيل نمطي الـ polling والـ Interrupt من خلال التطبيق العملي.

#### 4- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ Polling :

- دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling: سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:

##### 1. دالة الإرسال:

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

##### حيث:

**huart**: وهو مؤشر يشير إلى Struct\_UART\_HandleTypeDef أي المنفذ التسلسلي المستخدم للاتصال مثلاً قد يكون &huart1 أو &huart2 أو &huart3 أو &huart4 وهكذا....

**pData**: وهو مؤشر أيضاً يشير إلى البيانات التي سيتم إرسالها عبر UART وكما نرى نوعه uint8\_t أي يقبل إرسال بيانات من نوع Unsigned int وبطول 8bit، مثال: قد تكون pData مصفوفة وليكن اسمها Data وتكون معرفة بالشكل التالي:

```
uint8_t Data[] = {0,1,2,3,4,5,6,7,8,9};
```

##### حيث:

**Size**: وهو متغير يعبر عن حجم البيانات التي سيتم إرسالها أي حجم الـ pData وهي في المثال السابق 10.

**Timeout**: أقصى زمن يتم انتظاره بالميللي ثانية حتى يتم اكتمال عملية الإرسال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الإرسال سيتم قطع عملية الإرسال وتقوم الدالة بإرجاع HAL\_TIMEOUT ماعدا ذلك يتم إرجاع HAL\_OK، ويمكن استخدام الدالة HAL\_MAX\_DELAY مكان Timeout ووظيفتها انتظار أقصى زمن ممكن لعملية الإرسال.

مثال: إذا أردنا إرسال المصفوفة التالية عبر المنفذ التسلسلي الأول UART1:

```
/* USER CODE BEGIN 0 */
uint8_t data[]={0,1,2,3,4,5,6,7,8,9};
/* USER CODE END 0 */
```

نستخدم الدالة التالية:

```

/* USER CODE BEGIN 3 */
/* Infinite loop */
while (1)
{
    HAL_UART_Transmit(&huart1,data,10,1000);
}
/* USER CODE END 3 */

```

## 2. دالة الاستقبال:

إذا أردنا استقبال بيانات على UART باستخدام وضع Polling ومكتبات HAL نقوم باستدعاء الدالة التالية:

```

HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t
Timeout);

```

حيث:

**Size:** وهو متغير يعبر عن حجم البيانات التي سيتم استقبالها أي pData وهي في المثال التالي 10.

**Timeout:** أقصى زمن يتم انتظاره بالميلي ثانية حتى يتم اكتمال عملية الاستقبال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الاستقبال سيتم قطع عملية الاستقبال وتقوم الدالة بإرجاع HAL\_TIMEOUT ماعدا ذلك يتم إرجاع HAL\_OK، ويمكن استخدام الدالة HAL\_MAX\_DELAY مكان Timeout ووظيفتها انتظار أقصى زمن ممكن لعملية الاستقبال.

**مثال:** إذا أردنا استقبال مصفوفة أعداد صحيحة من المنفذ التسلسلي UART1:

```

/* USER CODE BEGIN 0 */
uint8_t data[10];
/* USER CODE END 0 */

```

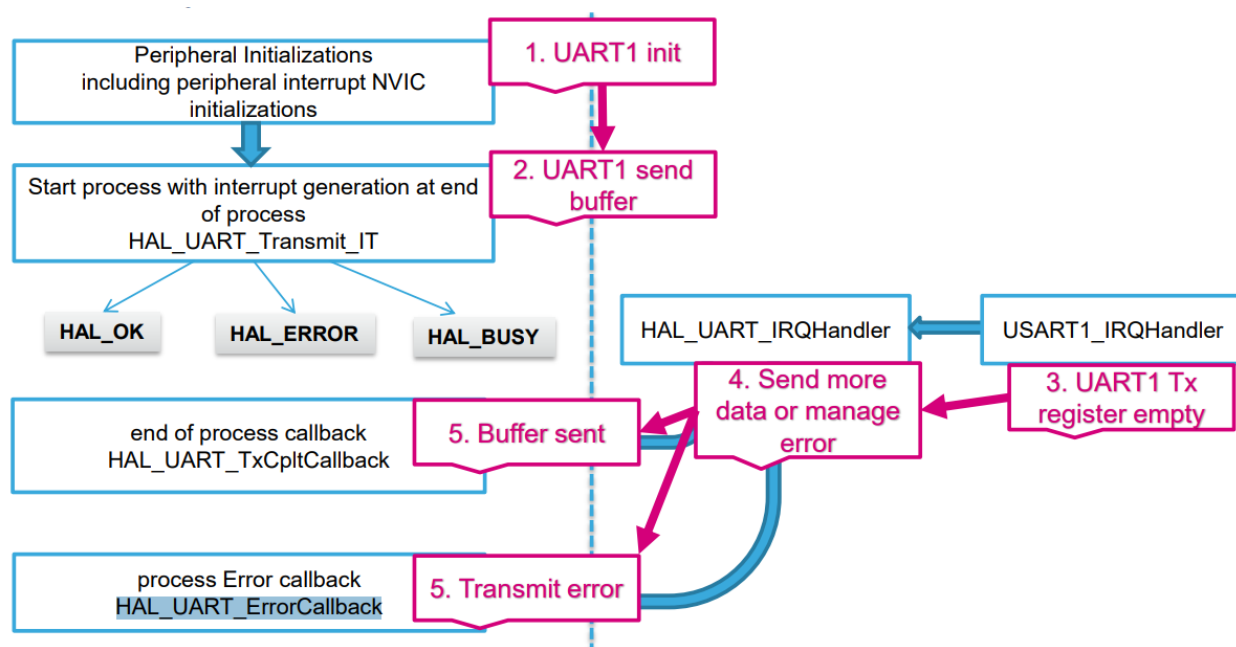
```

/* USER CODE BEGIN 3 */
/* Infinite loop */
while (1)
{
    HAL_UART_Receive(&huart1,data,10,1000);
}
/* USER CODE END 3 */

```

## 5- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt:

خطوات تفعيل واستخدام المقاطعة مع المنفذ التسلسلي: يتم تهيئة و استخدام المقاطعة مع المنفذ التسلسلي وفق المخطط التالي:



### الخطوة الأولى:

تهيئة المنفذ التسلسلي من خلال اختيار رقم المنفذ التسلسلي وضبط معدل نقل البيانات BaudRate وتفعيل المقاطعة على المنفذ وغيرها من الإعدادات ويتم ذلك باستخدام أداة الـ CubeMx (التي أصبحت مدمجة داخل بيئة STM32CubeIDE)، وتلقائياً يتم إضافة سطور التهيئة عند توليد الكود ضمن ملف الـ main.c والملف stm32G0xx\_hal\_msp.c.

### الخطوة الثانية:

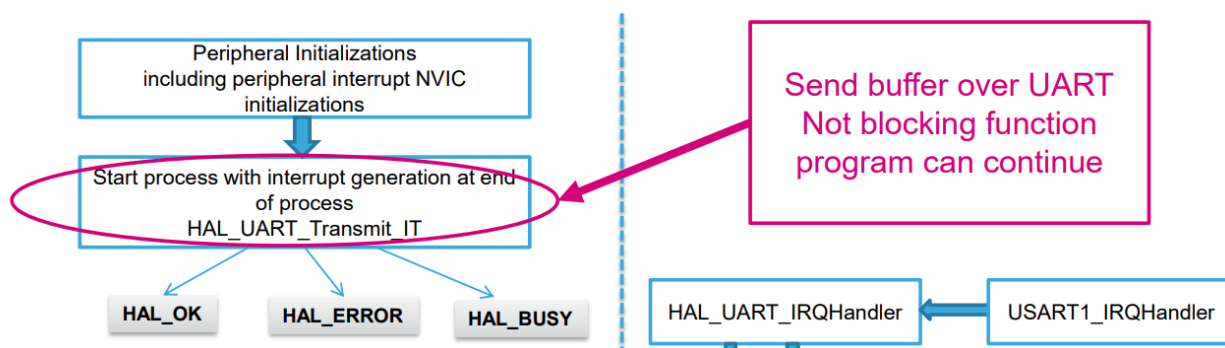
إرسال البيانات عبر المنفذ التسلسلي من خلال استخدام الدالة التالية:

```
HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```

حيث يتم إدخال بارامترات هذه الدالة كما قمنا بالشرح سابقاً، وكما تلاحظ فقد تم إضافة IT في اسم الدالة وأيضاً تم إزالة Timeout من بارامترات هذه الدالة مقارنةً بالدالة المستخدمة في نمط الـ Polling، لأنه لم يعد هناك زمن انتظار في نمط المقاطعة.

لاستقبال مجموعة من البايتات عبر المنفذ التسلسلي نستخدم الدالة التالية:

```
HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```



تعيد دالة الإرسال أو الاستقبال إما HAL\_OK في حال تمت عملية الإرسال/الاستقبال بنجاح، أو HAL\_error في حال حدوث خطأ أثناء عملية الإرسال/الاستقبال أو HAL\_Busy.

**الخطوة الثالثة:** تفحص فيما إذا أصبح مسجل البيانات UART1\_TX فارغاً

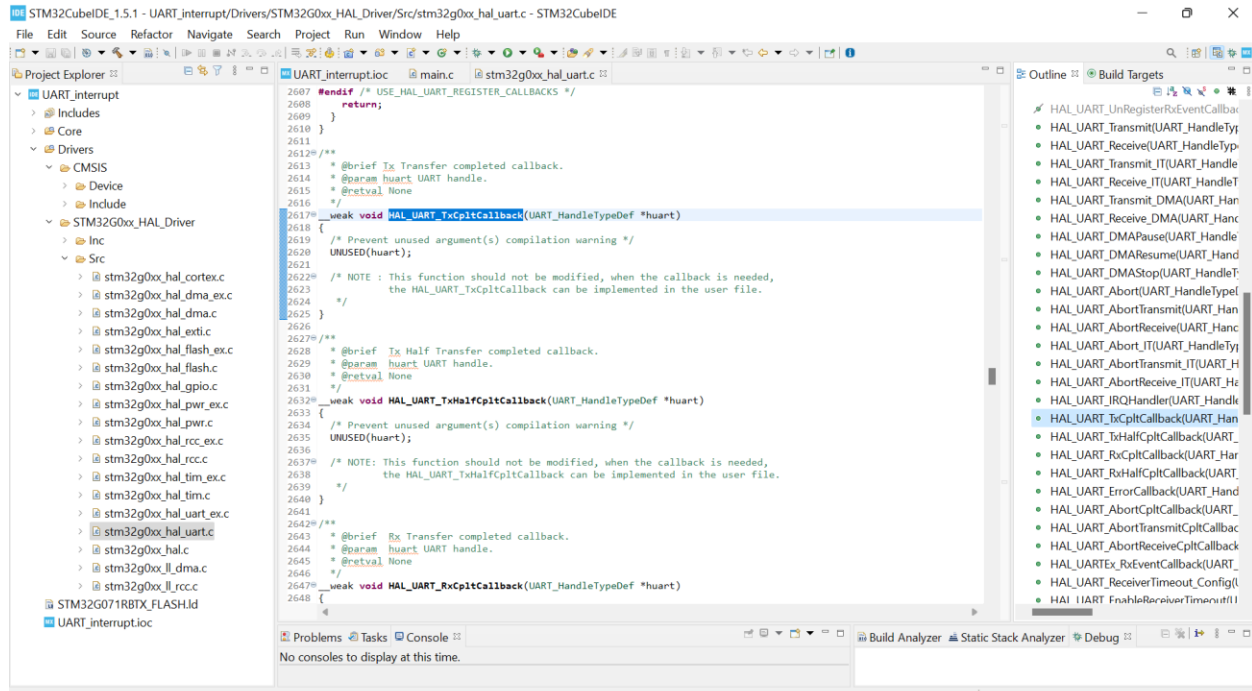
**الخطوة الرابعة:** في حال كان مسجل البيانات UART1\_TX فارغاً (يتم اكتشافه من خلال حدث المقاطعة وبالتالي استدعاء HAL\_UART\_IRQHandler والمعرف ضمن ملف stm32G0xx\_hal\_uart.c) فلدينا خيارين إما إرسال البيانات المتبقية في حال كان الـ Buffer غير فارغ أو الإعلان عن انتهاء عملية الإرسال، أو الكشف عن وجود خطأ في عملية الإرسال.



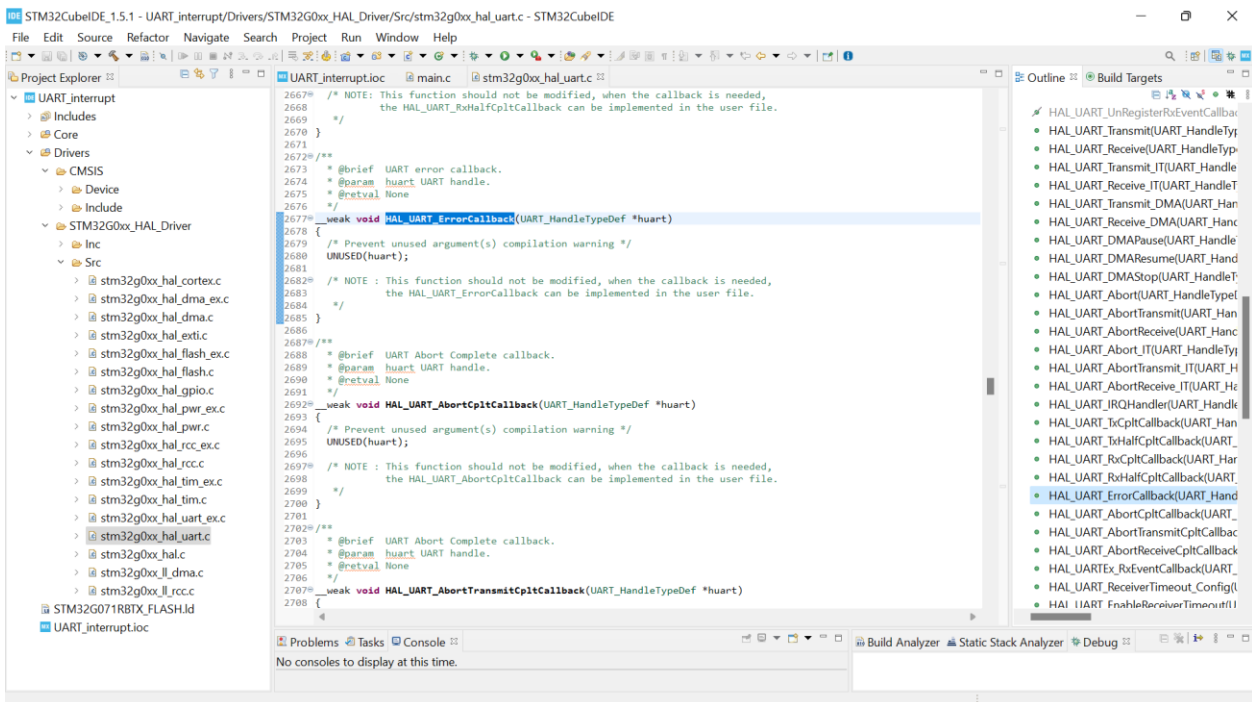
**الخطوة الخامسة:** لدينا خيارين:

في حال تم إرسال البيانات بنجاح عندها يتم استدعاء الدالة HAL\_UART\_TxCpltCallback والتي تكون معرفة كـ weak (أي يتم استدعائها في حال لم يتم إعادة تعريفها ضمن البرنامج الرئيسي) ضمن ملف stm32G0xx\_hal\_uart.c ضمن مجلد stm32G0xx\_HAL\_Driver ثم المجلد Src، نقوم بنسخ الدالة HAL\_UART\_TxCpltCallback ووضعها ضمن ملف البرنامج الرئيسي main.c لاستخدامها لاحقاً.

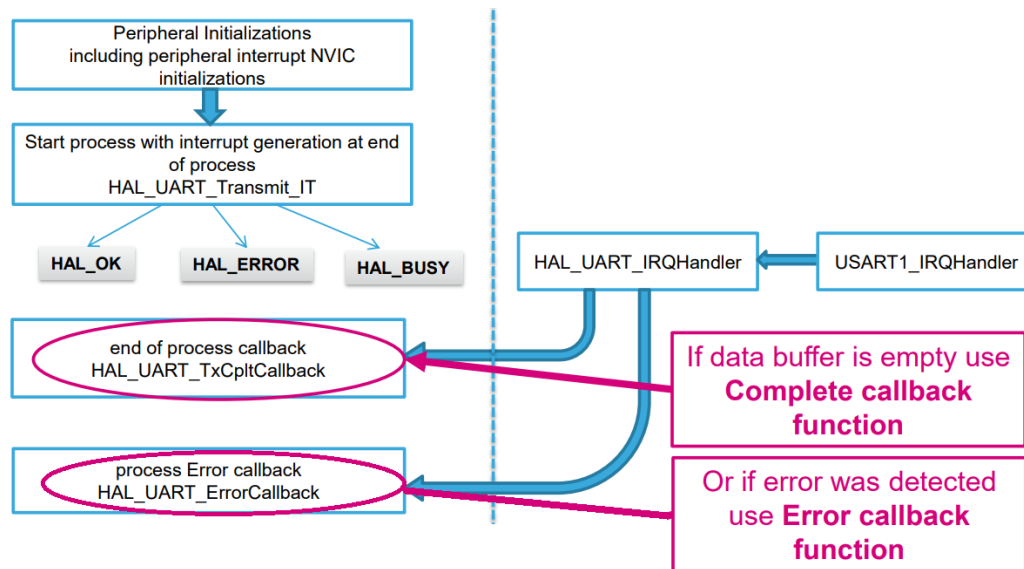




في حال تم اكتشاف خطأ أثناء إرسال البيانات يتم استدعاء الدالة HAL\_UART\_ErrorCallback والتي تكون معرفة كـ weak\_ (أي يتم استدعائها في حال لم يتم إعادة تعريفها ضمن البرنامج الرئيسي) ضمن ملف stm32G0xx\_hal\_uart.c ضمن مجلد stm32G0xx\_HAL\_Driver ثم المجلد Src كما في الشكل التالي:

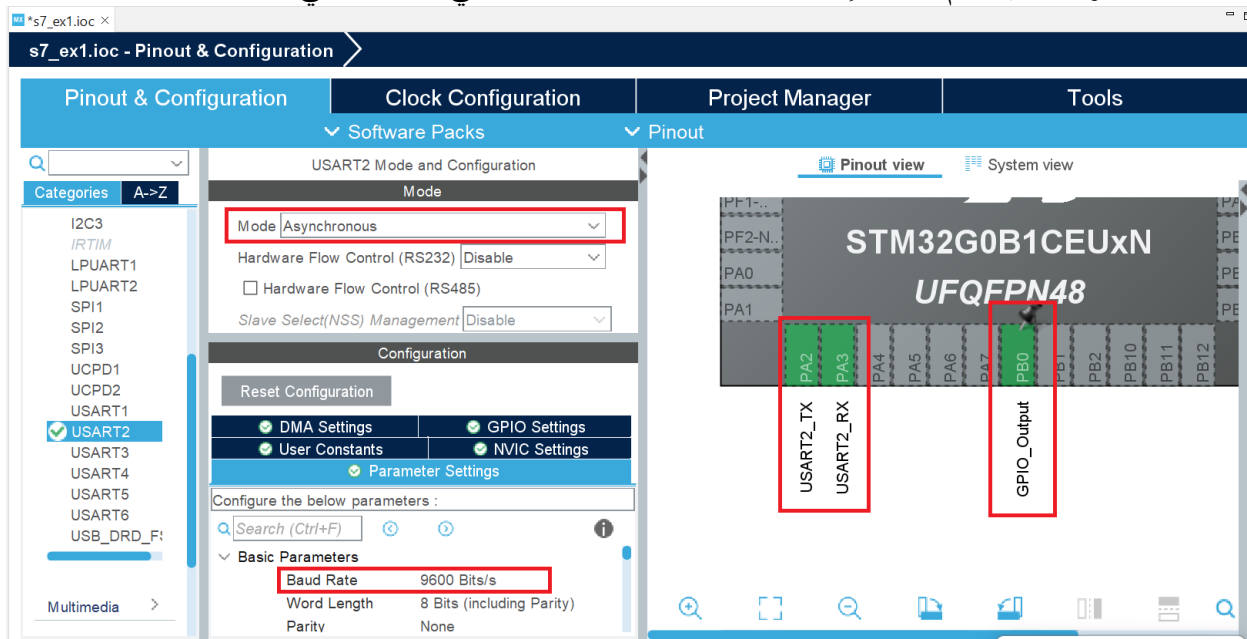


نقوم بنسخ الدالة HAL\_UART\_ErrorCallback ووضعها ضمن ملف البرنامج الرئيسي main.c لاستخدامها لاحقاً.



6- التطبيق العملي الأول: مراقبة عمل أي كود (Debugging) من خلال نافذة الاتصال التسلسلي UART وباستخدام نمط الـ Polling: للاتصال باللوحة من خلال الـ Serial نحتاج إلى دائرة FT232RL.

الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشاء مشروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر STM32G0B1CEU6N  
الخطوة الثانية: قم بضبط إعدادات المنفذ UART2 ، كما في الشكل التالي:



من ضمن الإعدادات الموجودة Basic Rate ما يلي:

**Baudrate:** يمثل معدل نقل البيانات بوحدة Bit/Sec بين المرسل والمستقبل، ولها قيم قياسية يتم الاختيار منها، حيث تعتمد هذه القيم على وحدة الـ Clock Peripheral الخاصة بالـ USART وهي عبارة عن ساعة المتحكم المصغر مقسومة على رقم ثابت، لكن ليس كل الـ BaudRates المتاحة يمكن استخدامها فقد ينتج عن المعدلات العالية أخطاء، حيث يوضح الشكل التالي الـ BaudRates القياسية والأخطاء التي قد تنجم عن كل BaudRate، حيث تختلف قيم الـ BaudRates المتاحة من متحكم لآخر بناءً على Clock Peripheral التي يوفرها للـ USART فقد يدعم متحكم BaudRates أكثر أو أقل.

Baud rate		Oversampling by 16		Oversampling by 8	
S.No	Desired (Bps)	Actual	%Error	Actual	%Error
2	2400	2400	0	2400	0
3	9600	9600	0	9600	0
4	19200	19200	0	19200	0
5	38400	38400	0	38400	0
6	57600	57620	0.03	57590	0.02
7	115200	115110	0.08	115250	0.04
8	230400	230760	0.16	230210	0.8
9	460800	461540	0.16	461540	0.16
10	921600	923070	0.16	923070	0.16
11	2000000	2000000	0	2000000	0
12	3000000	3000000	0	3000000	0
13	4000000	N.A.	N.A.	4000000	0
14	5000000	N.A.	N.A.	5052630	1.05
15	6000000	N.A.	N.A.	6000000	0

- اختر معدل نقل البيانات 9600 (عند استخدام مصدر الساعة الداخلية للمتحكم لا يمكن للمتحكم نقل البيانات عبر منفذ الـ UART بسرعة عالية)

**WordLength:** وتعني عدد البتات التي يتم إرسالها أو استقبالها في Frame (في المرة الواحدة)، وتوفر 3 قيم يمكن الاختيار بينها 7bit,8bit,9bit حيث لا يتضمن هذا الرقم البتات الخاصة بـ Start و Stop وغيرها.

**StopBits:** يحدد عدد البتات الخاصة بـ Stop التي سيتم إرسالها، ويمكن الاختيار بين 1 و 2 أي بت واحد أو 2bit في نهاية الإشارة.

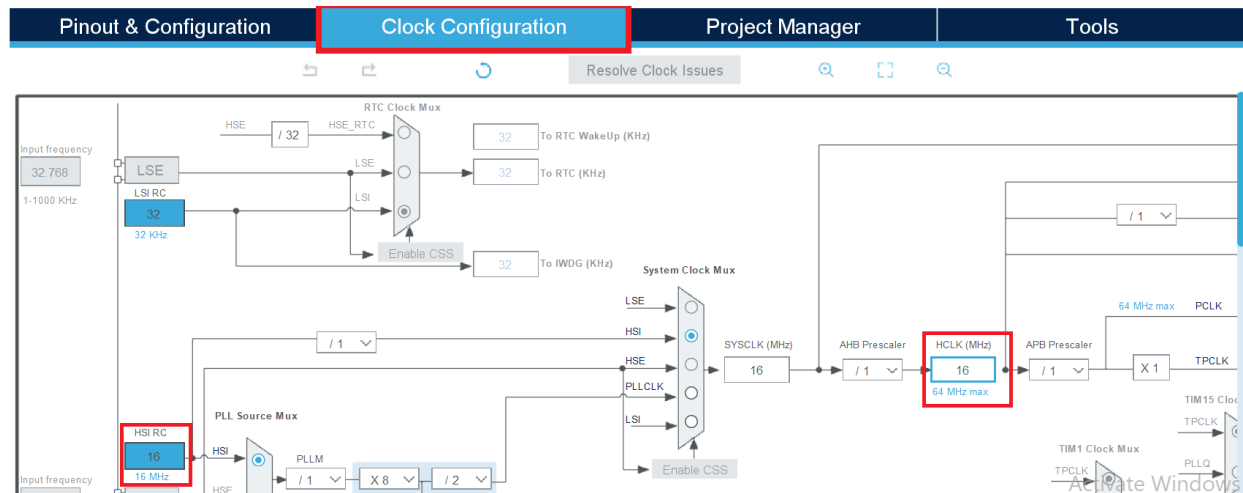
**Parity:** هو عبارة عن اختبار يستخدم لاكتشاف الأخطاء أثناء عملية الإرسال والاستقبال للبيانات من خلال الـ USART ، وهو عبارة عن بت يكون مكانه عند البت الأكثر أهمية MSB بحيث لو تم استخدام Word Length بـ 8-bit يكون مكانه في البت الثامن، أما لو تم استخدام 9-bit يكون مكانه هو في البت التاسع، ولها نمطين:

- فردي Odd: وتكون قيمة بت الـ Parity مساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إرسالها زوجي، وصفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إرسالها فردي.
- زوجي Even: وتكون قيمة بت الـ Parity مساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إرسالها زوجي، وصفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إرسالها فردي.

على سبيل المثال: عندما تريد إرسال أي بيانات يتم تحويلها للـ Binary فمثلاً إذا كنا نريد إرسال الكلمة التالية 0b01101110 فمن خلال الـ Parity يتم حساب عدد الواحدات الموجودة ضمن هذه الكلمة المراد إرسالها وهي في هذه الحالة 5، ففي حال كنت تستخدم نمط الفردي ستكون قيمة الـ Parity صفر، أما في حال كنت تستخدم نمط الزوجي ستكون قيمة الـ Parity واحد.

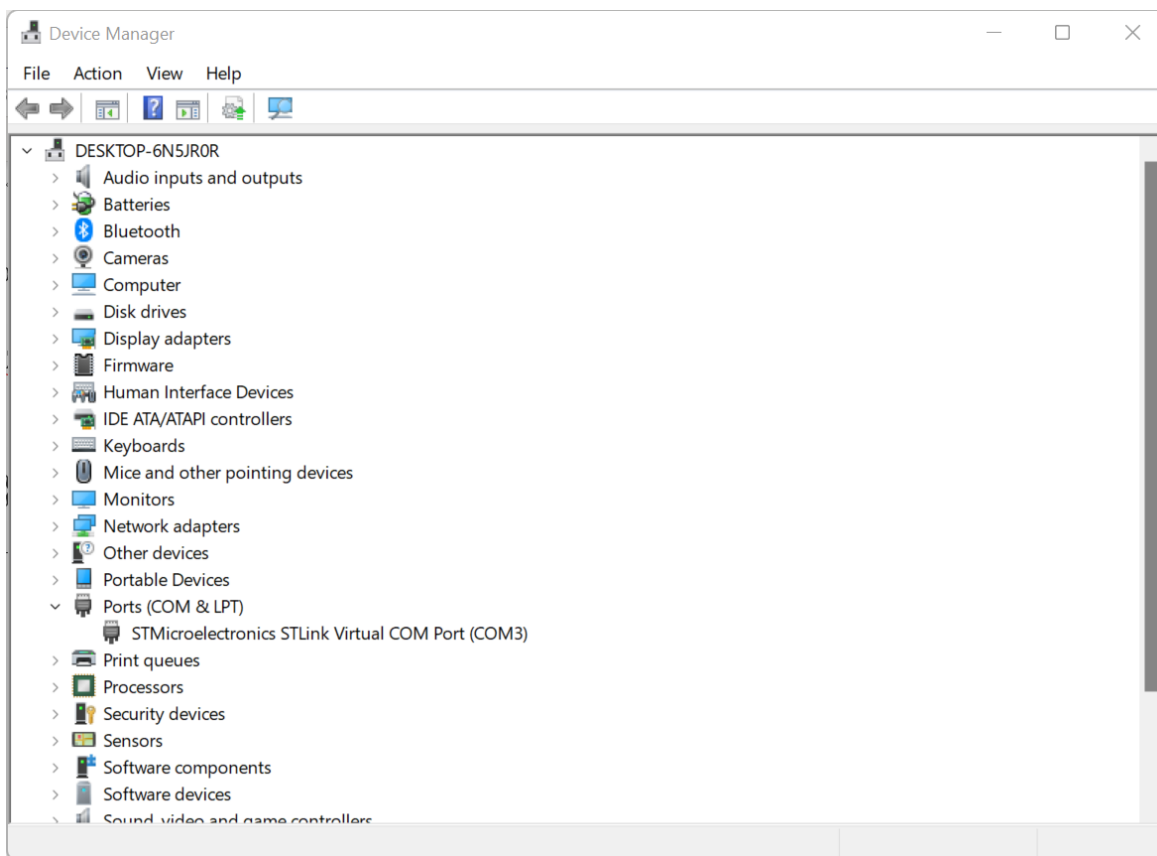
ويتم إرسال قيمة البت الخاص بالـ Parity من المرسل إلى المستقبل، فإن لم يحصل تطابق بين قيمته عند المرسل مع قيمته عند المستقبل فهذا يعني وجود خطأ ما في الإرسال حيث يتم طلب إعادة الإرسال.

**الخطوة الثالثة:** ضبط تردد الساعة للمتحكم سنختار مصدر الساعة الداخلي HSI:



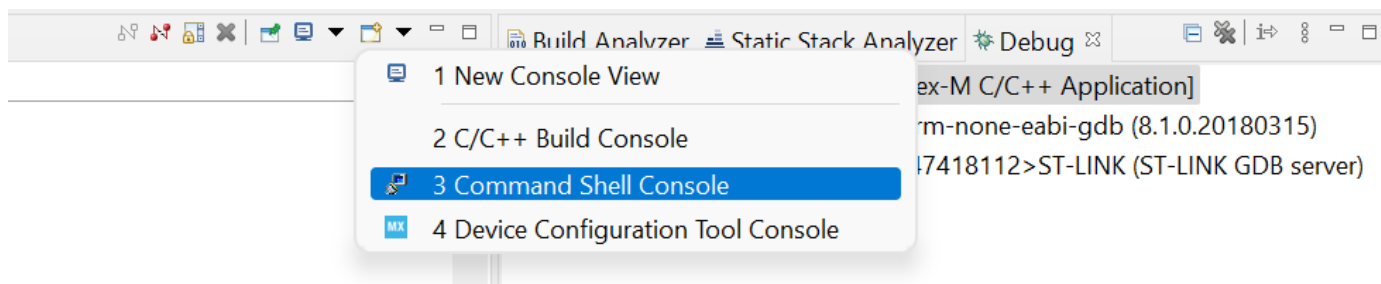
<pre> SystemClock_Config();  MX_GPIO_Init();  MX_USART2_UART_Init();  while (1) {     sprintf(MSG, "Hello <u>Dudes</u>! Tracing X = %d \n", X);      HAL_UART_Transmit(&amp;huart2,(uint8_t*) MSG, sizeof(MSG), HAL_MAX_DELAY);     HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);     HAL_Delay(500);     X++; }}</pre>	<p>استدعاء الدالة المسؤولة عن ضبط إعدادات ساعة المتحكم</p> <p>استدعاء الدالة المسؤولة عن ضبط أقطاب الدخل والخرج للمتحكم</p> <p>استدعاء الدالة المستخدمة لضبط إعدادات المنفذ التسلسلي للمتحكم</p> <p>حلقة While(1) اللانهائية</p> <p>تشكيل الجملة المراد إرسالها على المنفذ التسلسلي</p> <p>إرسال السلسلة MSG إلى المنفذ التسلسلي UART2</p> <p>إضافة تأخير زمني بين عمليات الإرسال المتكررة بمقدار 500 ميلي ثانية</p> <p>زيادة المتحول X في كل مرة</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

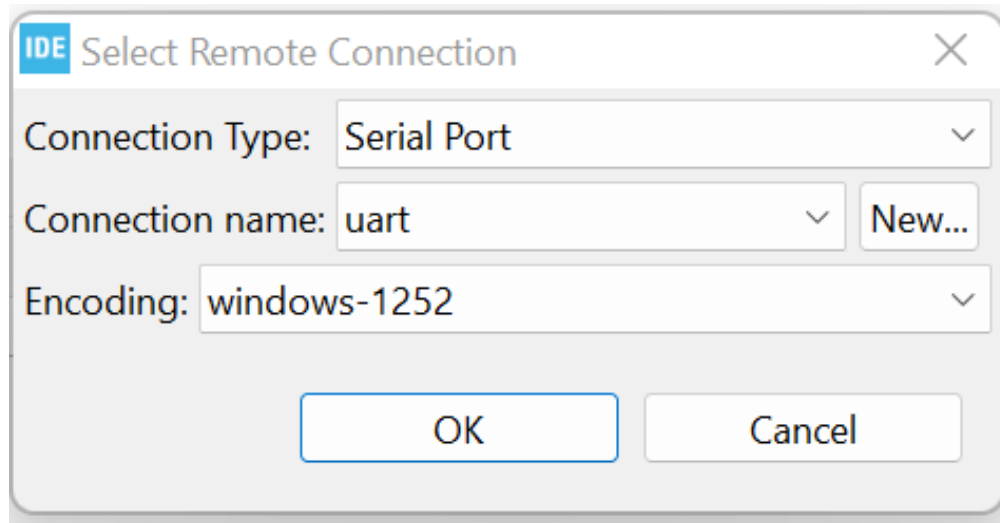
**الخطوة السادسة:** ترجمة الكود وإرساله إلى البورد التطويري ، ثم قم بفتح Device Manager لتعرف رقم المنفذ التسلسلي المستخدم من الحاسب للاتصال مع اللوحة:



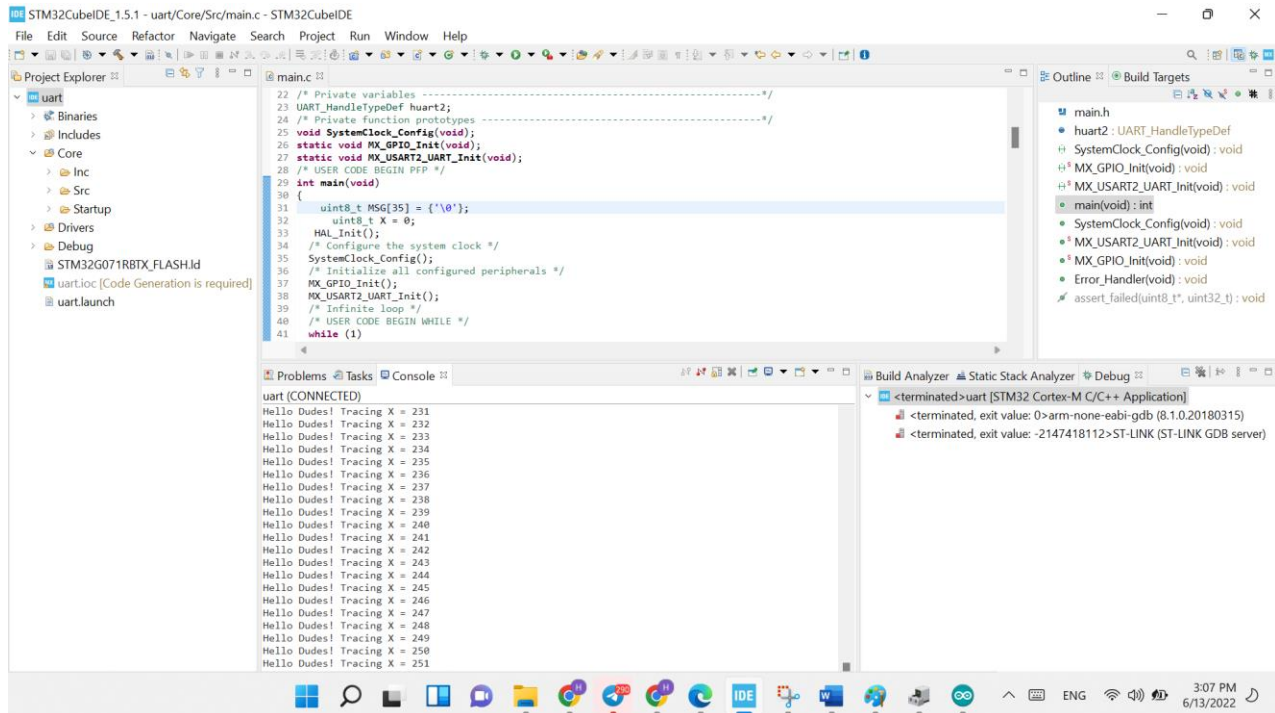
**الخطوة السابعة:** من القائمة Windows اختر show view ثم console فتجد أن الـ console ظهرت في أسفل الشاشة ، ثم من console نختار command shell console ثم نختار نوع منفذ الاتصال وهو Serial port

ونعطيه اسم مثلاً uart كما في الشكلين التاليين:





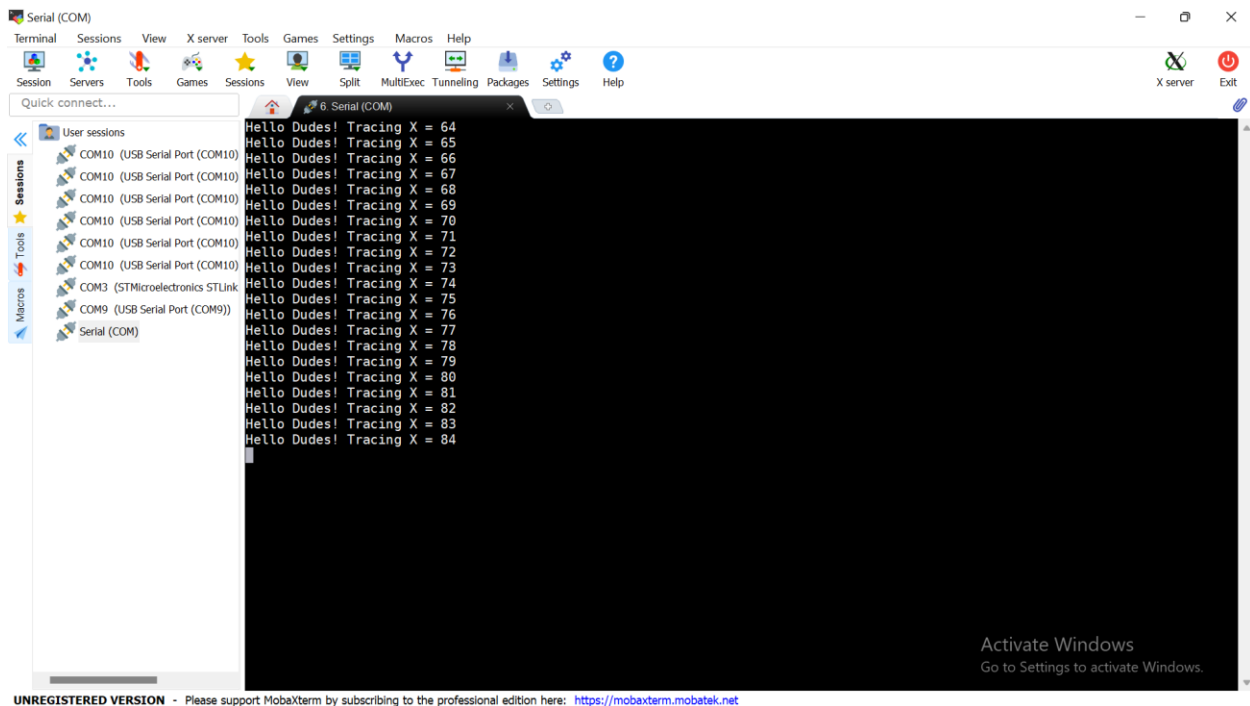
فتبدأ عملية طباعة الجملة المرادة كما في الشكل التالي:



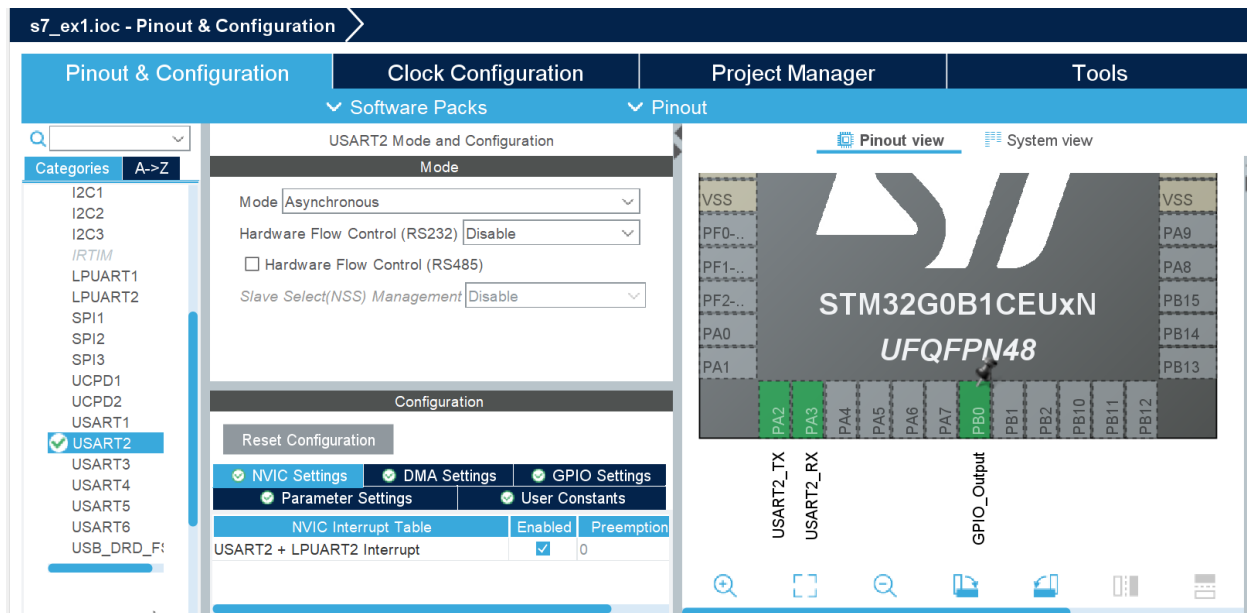
ملاحظة:

بإمكانك استخدام أي أداة للتخاطب مع المنفذ التسلسلي على سبيل المثال Mobaxterm أو حتى من خلال بيئة Arduino بعد ضبط الإعدادات الأساسية مثل رقم المنفذ التسلسلي للحاسب المتصل باللوحة وأيضاً معدل نقل البيانات BaudRate كما في الشكل التالي:





7- تطبيق عملي لاستخدام المنفذ التسلسلي USART من خلال نمط الـ interrupt:  
قم بإعادة الخطوتين الأولى والثانية كما في التطبيق السابق ثم قم بتفعيل المقاطعة للمنفذ التسلسلي



**الخطوة الثالثة:** نقوم بضبط إعدادات ساعة النظام كما في التطبيق السابق، ثم قم بتوليد الكود من خلال الضغط على `ctrl+s` فيتم بتوليد الكود آلياً.

**الخطوة الرابعة:** قم بكتابة الكود بالشكل التالي (طبعاً هناك قسم كبير من الكود تم توليده من خلال MX وقمنا نحن فقط بإكمال الكود بما يناسب التطبيق)

<pre>#include "main.h"  UART_HandleTypeDef huart2;  uint8_t MSG[35] = {'\0'}; uint8_t X = 0;  void SystemClock_Config(void);  static void MX_GPIO_Init(void);</pre>	<p>المنفذ التسلسلي الذي سيتم استخدامه هو UART2 وتم تسميته بـ huart2</p> <p>تصريح عن مصفوفة محارف بعدد 35 محرف</p> <p>تصريح عن متحول ليعمل كعداد من نوع uint_8t وإسناد قيمة صفرية له كقيمة ابتدائية</p> <p>تعريف الدالة الخاصة بتهيئة ساعة النظام</p> <p>تعريف الدالة الخاصة بتهيئة أقطاب الدخل والخرج</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```

static void MX_USART2_UART_Init(void);
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_USART2_UART_Init();

    sprintf(MSG, "Hello Dudes! Tracing X =
%d\r\n", X);
    HAL_UART_Transmit_IT(&huart2,MSG,
    sizeof(MSG));

    while (1)
    {
        X++;
        HAL_Delay(100);
    }
}

void
HAL_UART_TxCpltCallback(UART_HandleTypeDef
ef *huart)
{
    HAL_GPIO_TogglePin(GPIOB,
    GPIO_PIN_0) ;
    sprintf(MSG, "Hello Dudes! Tracing
X = %d\r\n", X);
    HAL_UART_Transmit_IT(&huart2,
    MSG, sizeof (MSG));
}

```

تعريف الدالة الخاصة بتهيئة  
المنفذ التسلسلي  
بداية البرنامج الرئيسي  
تهيئة مكتبة HAL

تشكيل الجملة المراد إرسالها  
على المنفذ التسلسلي  
إرسال السلسلة MSG إلى  
المنفذ التسلسلي UART2

زيادة المتحول X في كل مرة  
إضافة تأخير زمني بين  
عمليات الإرسال المتكررة  
بمقدار 100 ميلي ثانية

الدالة التي يتم استدعائها عند  
إتمام عملية إرسال  
البيانات (قمنا بنسخ اسمها من  
stm32g0xx\_hal\_uart.c)  
(حيث تكون معرفة كـ  
(weak  
لاحظ أننا قمنا بتغيير الحالة  
المنطقية لليد وإعادة إرسال  
البيانات مجدداً

<pre>void HAL_UART_ErrorCallback(UART_HandleTypeDef f *huart) {     __NOP(); }</pre>	<p>الدالة التي يذهب إليها المعالج في حال لم تتم عملية الاستقبال بشكل صحيح لأي سبب من الأسباب</p> <p>وضعنا هذه التعليمة فقط من أجل الـ debug.</p>
--------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------