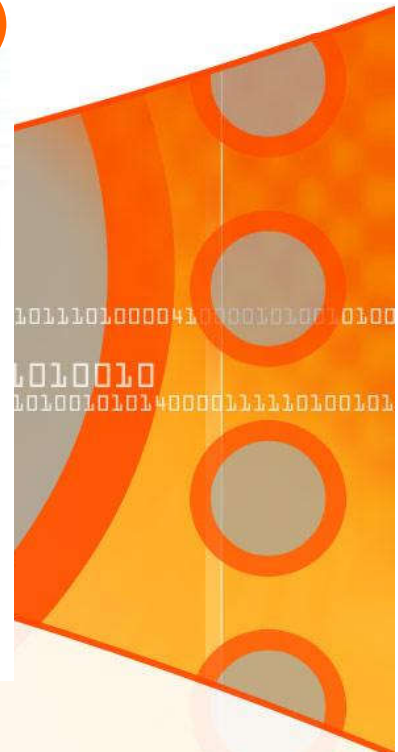
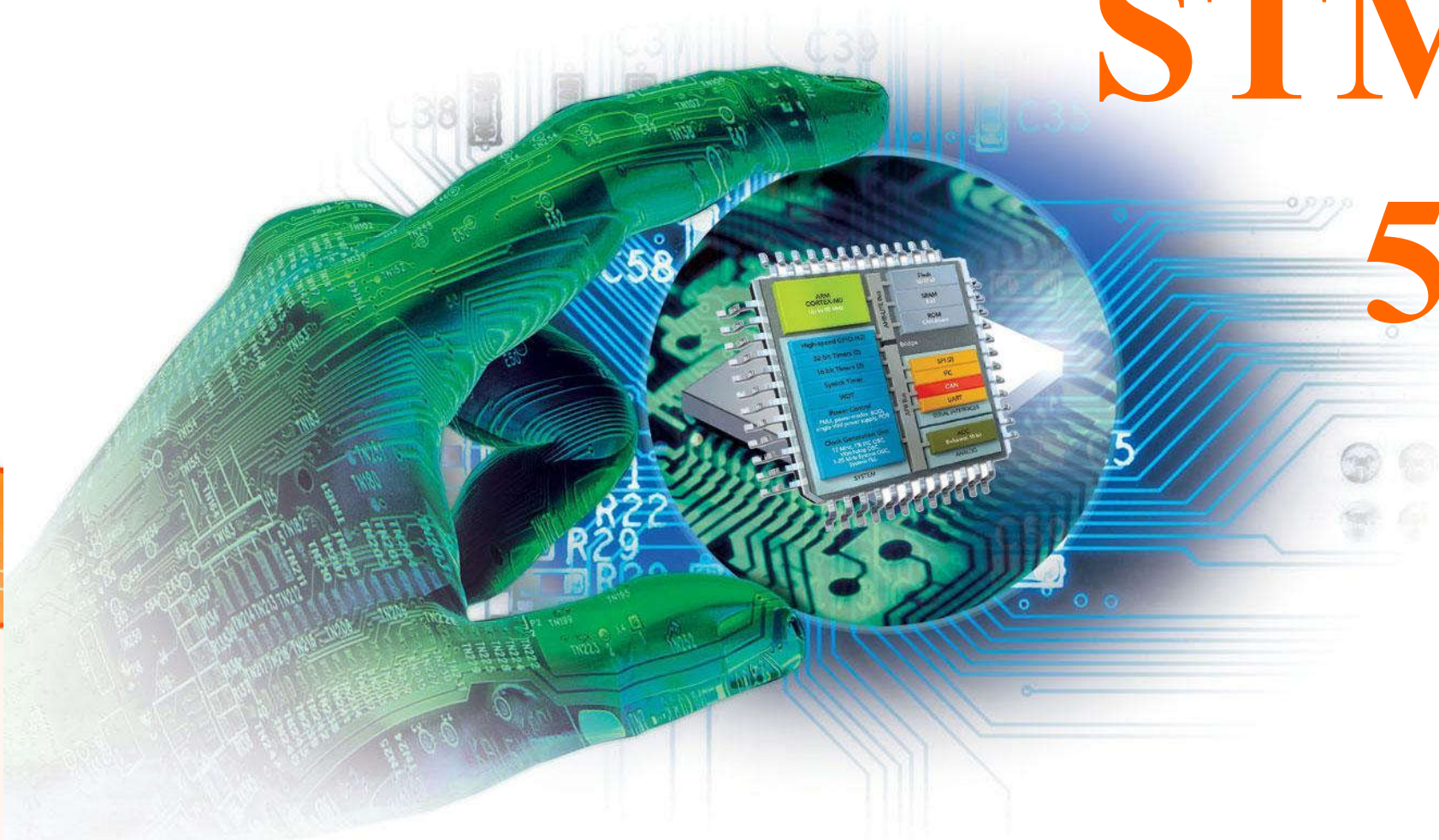


# متحكمات

# STM32

# 5



- ☐ مقارنة بين بروتوكولات الاتصال التسلسلي الشائعة
- ☐ مفهوم UART و USART
- ☐ أنماط العمل المختلفة للـ UART في متحكمات STM32
- ☐ دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ Polling
- ☐ تطبيق عملي لاستخدام المنفذ التسلسلي USART من خلال نمط الـ polling
- ☐ إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt
- ☐ دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في نمط الـ interrupt
- ☐ تطبيق عملي لاستخدام المنفذ التسلسلي USART من خلال نمط الـ interrupt

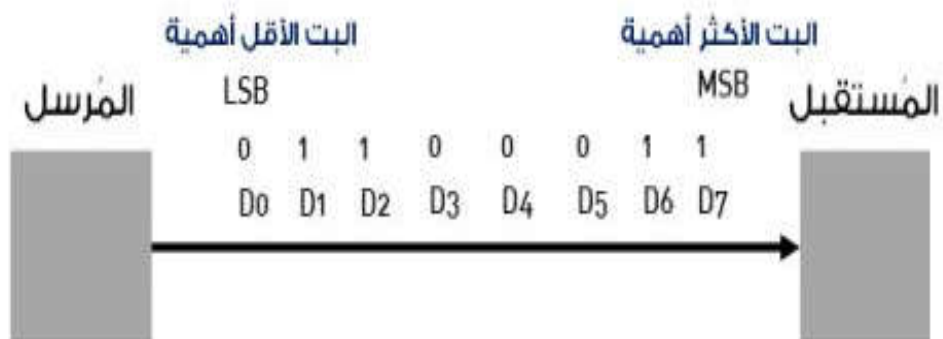
# مقارنة بين بروتوكولات الاتصال التسلسلي الشائعة:

	RS232	RS485	I2C	SPI	M-wire	1-wire	USB	CAN
Sync/Async	Async	Async.	Sync.	Sync.	Sync.	Async.	Async.	Async.
Type	peer-peer	master/slaves	multi-master	multi-master	master/slaves	master/slaves	host/device	multi-master
Duplex	full	half	half	full	full	half	half	half
Signaling	single-ended	Differential	single-ended	single-ended	single-ended	single-ended	Differential	Differential
Max Devices No.	2	32, 128, 256	40 (cap=400pf)	8 (cap, circuit)	8 (cap, circuit)	20 (cap, power)	127 per controller	2048
Data Rate	Up to 115Kbps	Up to 35Mbps	Std.: 100kbps Fast: 400kbps Hi: 3.4Mbps	Up to 10Mbps	Up to 1Mbps	Std.: 16.3Kbps Overdrive: 142kbps	Low: 1.5Mbps Full: 12Mbps Hi : 480Mbps	Up to 1Mbps
Max. Length	15m	1200m (at 100kbps)	6m	3m	3m	300m	5m	1000m (at 62kbps)
Pin Count	2* (Tx, Rx)	2 (A, B)	2 (SDA, SCL)	3 + SS* (SI, SO, SCK)	3 + SS* (DI, DO, SK)	1 (IO)	2 (A+, A-)	2 (CAN_H, _L)
Interfacing	HW	HW	SW   HW	HW   SW	HW   SW	HW & SW	protocol stack	HW & SW
Flow Control	HW or SW handshake	HW or SW handshake	Acknowledge from slave	None	None	CRC, Pulling	Polling by controller	CSMA / CDAMP

# مقارنة بين بروتوكول الاتصال التسلسلي والتفرعي:



في الاتصال التفرعي يمكن إرسال عدة بتات رقمية بنفس اللحظة الزمنية، ما يمنح سرعة كبيرة في نقل البيانات. بنفس الوقت، تتطلب عملية الاتصال التفرعي ضمان توافق ساعة المرسل والمستقبل والتأكد من عدم تشويش نواقل الاتصال على بعضها البعض.



في الاتصال التسلسلي يتم إرسال البتات الرقمية الواحد تلو الآخر مع كل نبضة ساعة، ما يعني معدل أبطأ لنقل البيانات، ولكنها أقل تطلباً فيما يتعلق بضرورات توافق الساعة وتشويش نواقل الاتصال.



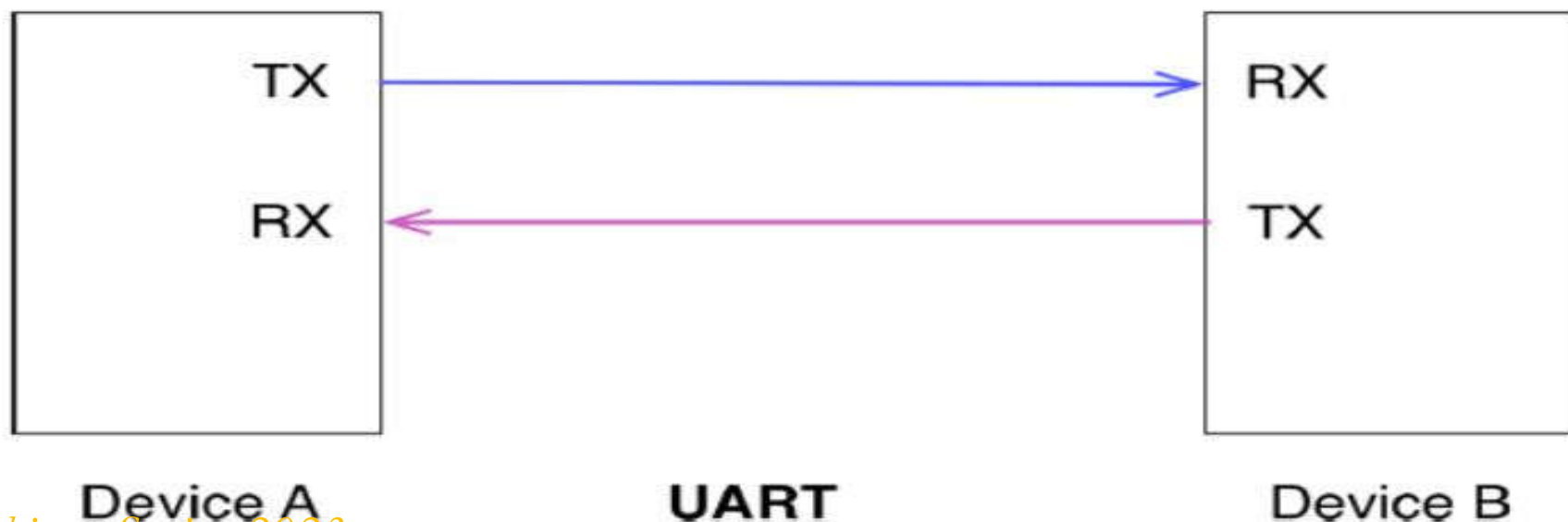
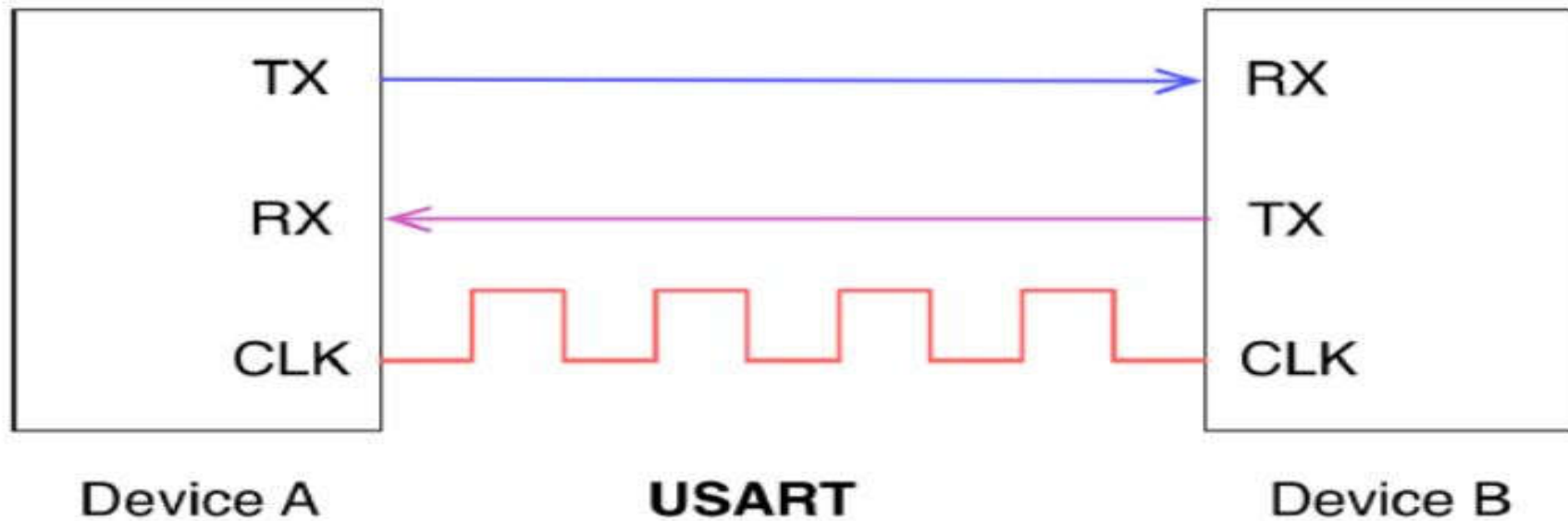
# النافذة التسلسلية USART/UART

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

**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.

# النافذة التسلسلية USART/UART

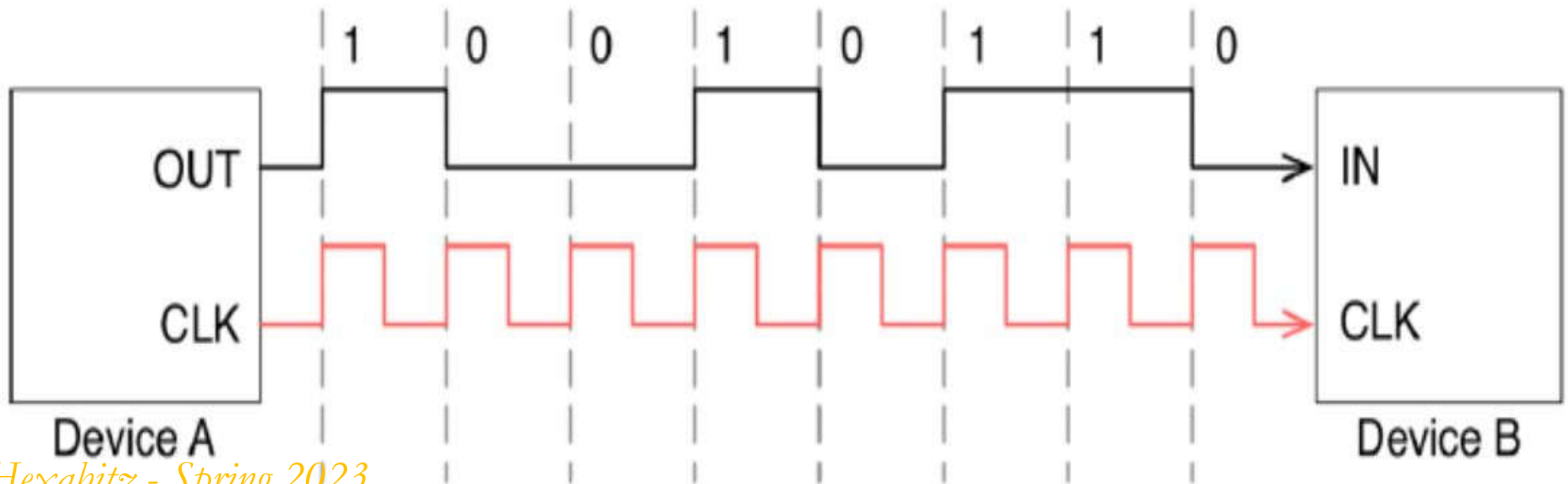


# USART

**U**niversal **S**ynchronous and **A**synchronous  
serial **R**eciever and **T**ransmitter

# النافذة التسلسلية USART

المخطط الزمني لعملية الإرسال التسلسلي المتزامن □  
Synchronous لبايت واحد 0 b01101001 من الجهاز  
Device A إلى الجهاز Device B حيث تم استخدام Clock  
لضبط توقيت إرسال البيانات، حيث يتم إرسال بت واحد مع كل  
جبهة صاعدة للـ Clock ، حيث تتعلق سرعة نقل البيانات بتردد الـ  
Clock ، فكلما زاد تردد الـ Clock كلما زادت سرعة نقل البيانات.

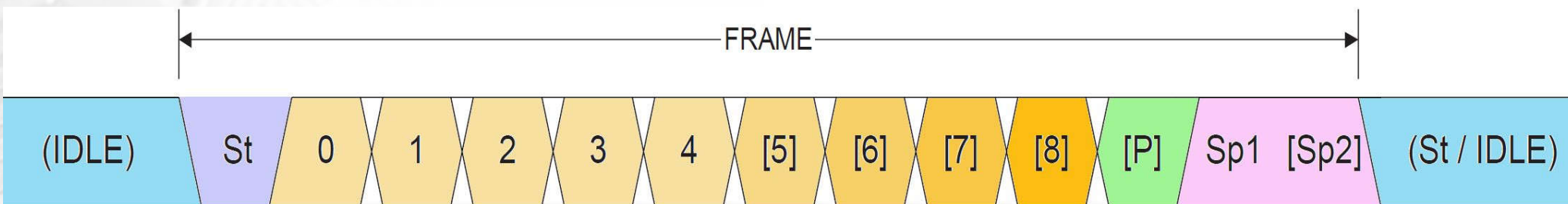




# النافذة التسلسلية UART

## (Interface Universal Asynchronous Receiver and Transmitter)

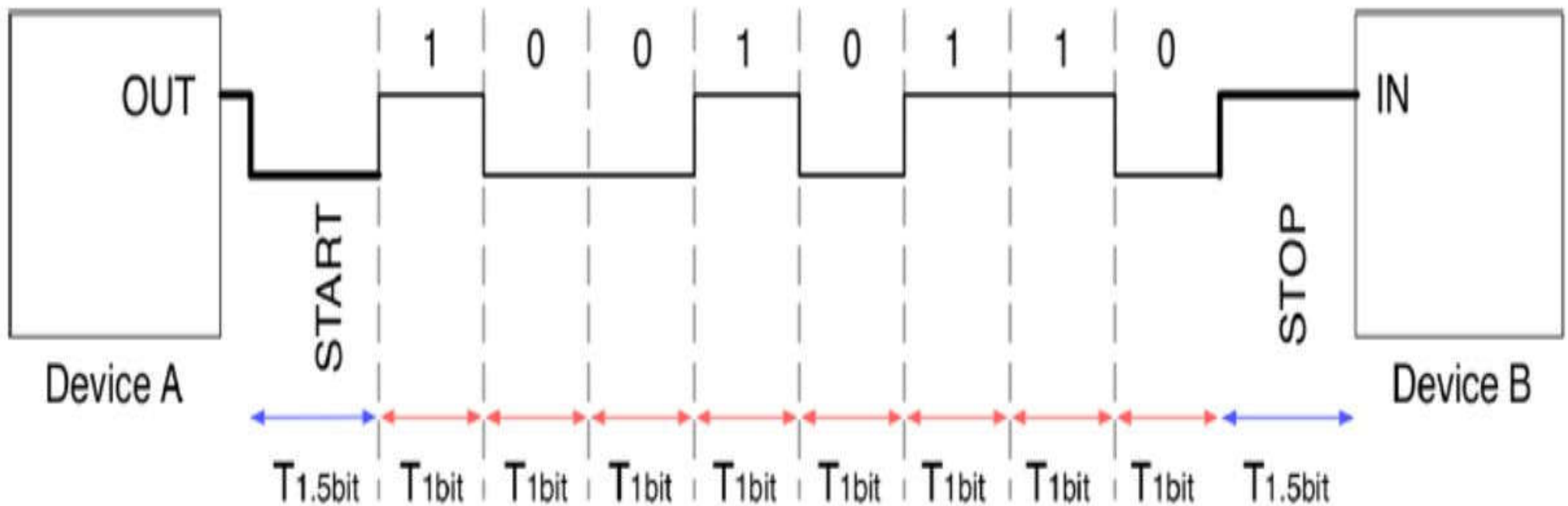
تعتبر هذه النافذة من أكثر نوافذ الاتصال التسلسلي استخداماً في الأنظمة الرقمية والمتحكمات المصغرة والطرفيات التي يمكن أن تربط معها...



## بنية إطار البيانات (UART Frame Format)

# النافذة التسلسلية UART

□ 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



□ تمثل الحالة الخاملة Idle state بإشارة HIGH وهي حالة عدم الإرسال

□ بداية الإرسال تتم من خلال Start Bit وتمثل بإشارة LOW

□ هذه الإشارة يتم اكتشافها من قبل المستقبل وتستغرق وقت  $1.5T$  حيث أن  $T$  هو الدور وهو عبارة عن مقلوب التردد أو ما يسمى Baud Rate أي معدل نقل البيانات بين المرسل والمستقبل

□ بعد ذلك يتم إرسال الـ 8bit والتي تمثل البيانات المراد إرسالها حيث يتم إرسال البت الأقل أهمية أولاً LSB ، وأحياناً يتم استخدام Parity Bit للتأكد من خلو البيانات من الأخطاء ويتم إنهاء الإرسال بـ Stop Bit

# أنماط العمل المختلفة للـ UART في متحكمات STM32

□ 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 ويستخدم عندما تكون كمية البيانات المتبادلة ليست كبيرة نسبياً ولا تمثل أهمية عالية من ناحية المعالجة

□ 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 Bps

# أنماط العمل المختلفة للـ UART في متحكمات STM32

□ 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



## مفاهيم أساسية في الاتصالات التسلسلية:

هناك بارامترات يجب تحديدها بين المرسل والمستقبل قبل إرسال البيانات في الاتصالات غير المتزامنة وهي:

- ✓ معدل سرعة الإرسال (Baud Rate): 1200, 2400, 9600...
- ✓ خانة فحص الإيجابية (Parity Bit): Even, Odd, or None.
- ✓ عدد البتات (Data Bits): 6, 7, 8, or 9-bit.
- ✓ عدد بتات التوقف (Stop Bit): 1 or 2.

# مفاهيم أساسية في الاتصالات التسلسلية:

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

# مفاهيم أساسية في الاتصالات التسلسلية:

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

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

## مفاهيم أساسية في الاتصالات التسلسلية:

□ 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 مساو 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 مساو للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إرسالها فردي، وصفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إرسالها زوجي.

# مفاهيم أساسية في الاتصالات التسلسلية:

على سبيل المثال:

□ عندما تريد إرسال أي بيانات يتم تحويلها للـ Binary فمثلاً إذا كنا نريد إرسال الكلمة التالية 0 b01101110 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 صفراً، أما في حال كنت تستخدم نمط الزوجي ستكون قيمة الـ Parity واحداً.

□ يتم 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.



# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling


سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:  
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 المستخدم للاتصال مثلاً قد يكون huart1& أو huart2& أو huart3& أو huart4&

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:  
دالة الإرسال: 

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

حيث:

 **Data** : 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}
```

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling


سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:  
دالة الإرسال: ☐

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

حيث:

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

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:  
دالة الإرسال: 

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

حيث:

**Timeout:**  أقصى زمن يتم انتظاره بالميلي 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 وهي وظيفتها انتظار أقصى زمن ممكن لعملية الإرسال

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

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 */
```



# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

□ 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:

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

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

□ دالة الاستقبال:

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

حيث:

□ **Data** : وهو مؤشر أيضاً يشير إلى البيانات التي سيتم استقبالها عبر UART وكما نرى نوعه `uint8_t` أي يقبل استقبال بيانات من نوع `Unsigned int` وبطول 8bit، مثال: قد تكون `pData` مصفوفة وليكن اسمها `Data` وتكون معرفة بالشكل التالي:

```
;Uint8_t Data[10]
```

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

□ دالة الاستقبال:

```
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 بالميللي 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 ووظيفتها انتظار أقصى زمن ممكن لعملية الاستقبال.

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Polling

□ دالة الاستقبال:

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

مثال:

```
/* 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 */
```

# إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

توفر جميع متحكمات STM32 مقاطعات لـ UART كما في الجدول التالي:

Interrupt Event	Event Flag	Enable Control Bit
Transmit Data Register Empty	TXE	TXEIE
Clear To Send (CTS) flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	RXNEIE
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error	NF or ORE or FE	FE EIE
Error in multi buffer communication		



# إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

يمكن تقسيم المقاطعات IRQs الخاصة بالمنفذ التسلسلي UART لمجموعتين:

□ IRQs : التي يتم استدعائها أثناء الإرسال:

- اكتمال الإرسال Transmission complete

- Clear to send (CTS)

- مسجل البيانات فارغ transmission Data Register Empty

- Noise Flag

- Framing error

# إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

IRQs : التي يتم استدعائها أثناء الاستقبال: 

Idle line detection •

Overrun error •

مسجل الاستقبال غير فارغ Receive data register not empty •

Parity error •

Lin break detection •

Noise Flag •


Framing error •

## إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt

□ يتم تفعيل حدث المقاطعة Interrupt event لكل نوع من خلال Enable Control Bit الخاص به كما في الجدول السابق، حيث كل هذه الـ IRQs لها فقط خط مقاطعة وحيد لكل USART Peripheral


□ باعتبار أن لكل وحدة USART في متحكمات STM32 خط مقاطعة وحيد لذا يتوجب على المستخدم تحليل علم المقاطعة Flag Event الذي تم رفعه لمعرفة المقاطعة التي حدثت

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ Interrupt

سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:  
دالة الإرسال: 

```
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 مقارنة بالدالة المستخدمة في نمط الـ Polling ، لأنه لم يعد هناك زمن انتظار في نمط المقاطعة.

# دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـ interrupt

□ دالة الاستقبال: إذا أردنا استقبال بيانات على UART باستخدام وضع interrupt ومكتبات HAL نقوم باستدعاء الدالة التالية:

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

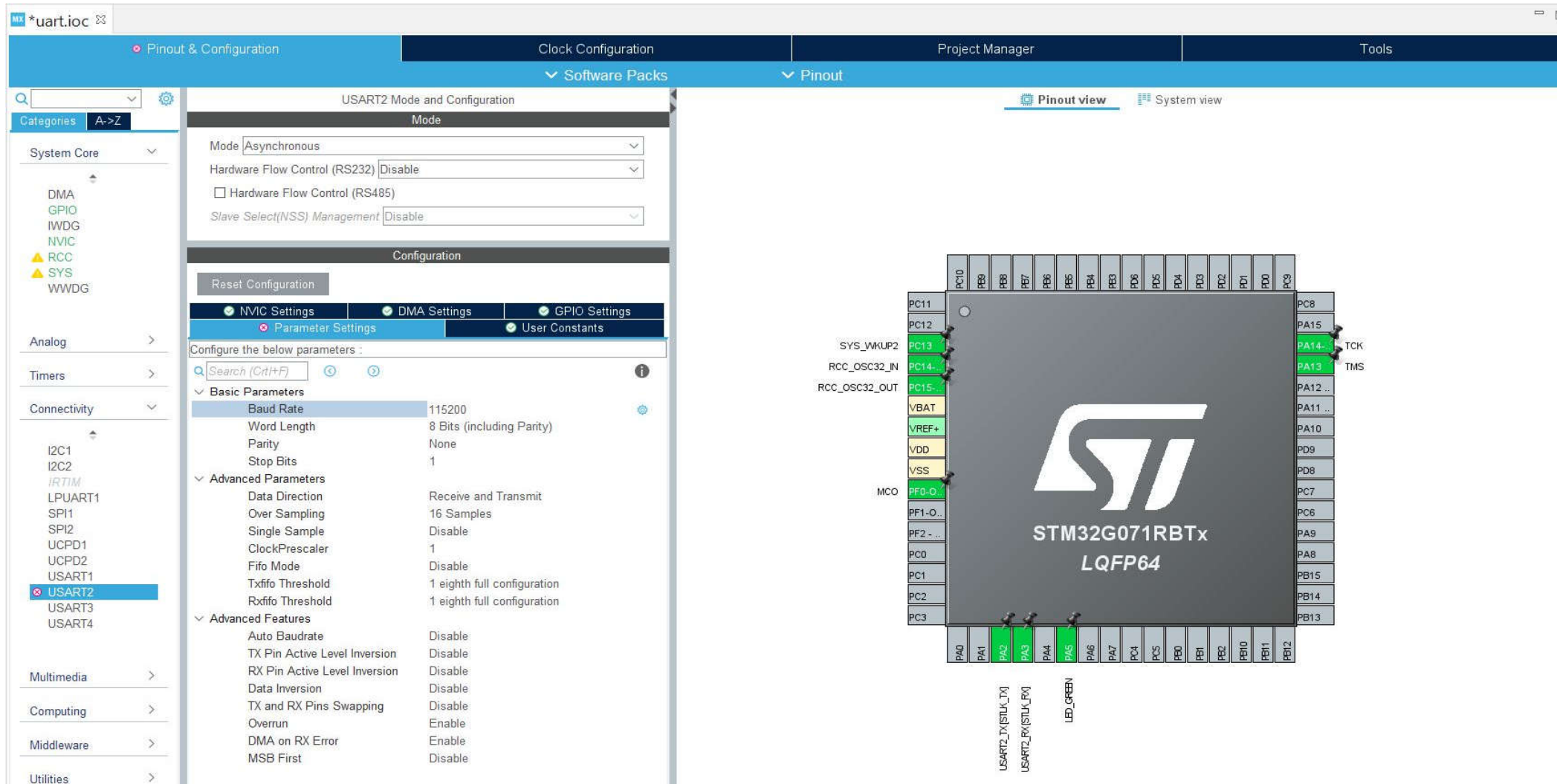
حيث:

□ 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



# التطبيق العملي 1: استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

نقوم بضبط إعدادات المنفذ التسلسلي: 



The screenshot displays the STM32CubeMX Pinout & Configuration window for the USART2 peripheral. The window is divided into several sections:

- Pinout & Configuration:** The main configuration area for the USART2 peripheral.
- Mode:** Asynchronous, Hardware Flow Control (RS232) Disabled, Slave Select (NSS) Management Disabled.
- Configuration:** Includes a Reset Configuration button and tabs for NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings (selected), and User Constants.
- Basic Parameters:** Baud Rate: 115200, Word Length: 8 Bits (including Parity), Parity: None, Stop Bits: 1.
- Advanced Parameters:** Data Direction: Receive and Transmit, Over Sampling: 16 Samples, Single Sample: Disable, Clock Prescaler: 1, Fifo Mode: Disable, Tx/Rx Threshold: 1 eighth full configuration.
- Advanced Features:** Auto Baudrate: Disable, TX Pin Active Level Inversion: Disable, RX Pin Active Level Inversion: Disable, Data Inversion: Disable, TX and RX Pins Swapping: Disable, Overrun: Enable, DMA on RX Error: Enable, MSB First: Disable.

The Pinout view shows the physical pins of the STM32G071RBTx LQFP64 package. The pins are color-coded: green for digital I/O, yellow for analog I/O, and blue for power/ground. The USART2 pins are highlighted in green: PA2 (TX), PA3 (RX), and PA5 (LED\_GREEN).

# التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

يصبح الكود بالشكل التالي: 

```
"include "main.h#  
;UART_HandleTypeDef huart2  
;void SystemClock_Config(void)  
;static void MX_GPIO_Init(void)  
;static void MX_USART2_UART_Init(void)  
int main(void)  
{
```

# التطبيق العملي 1: استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

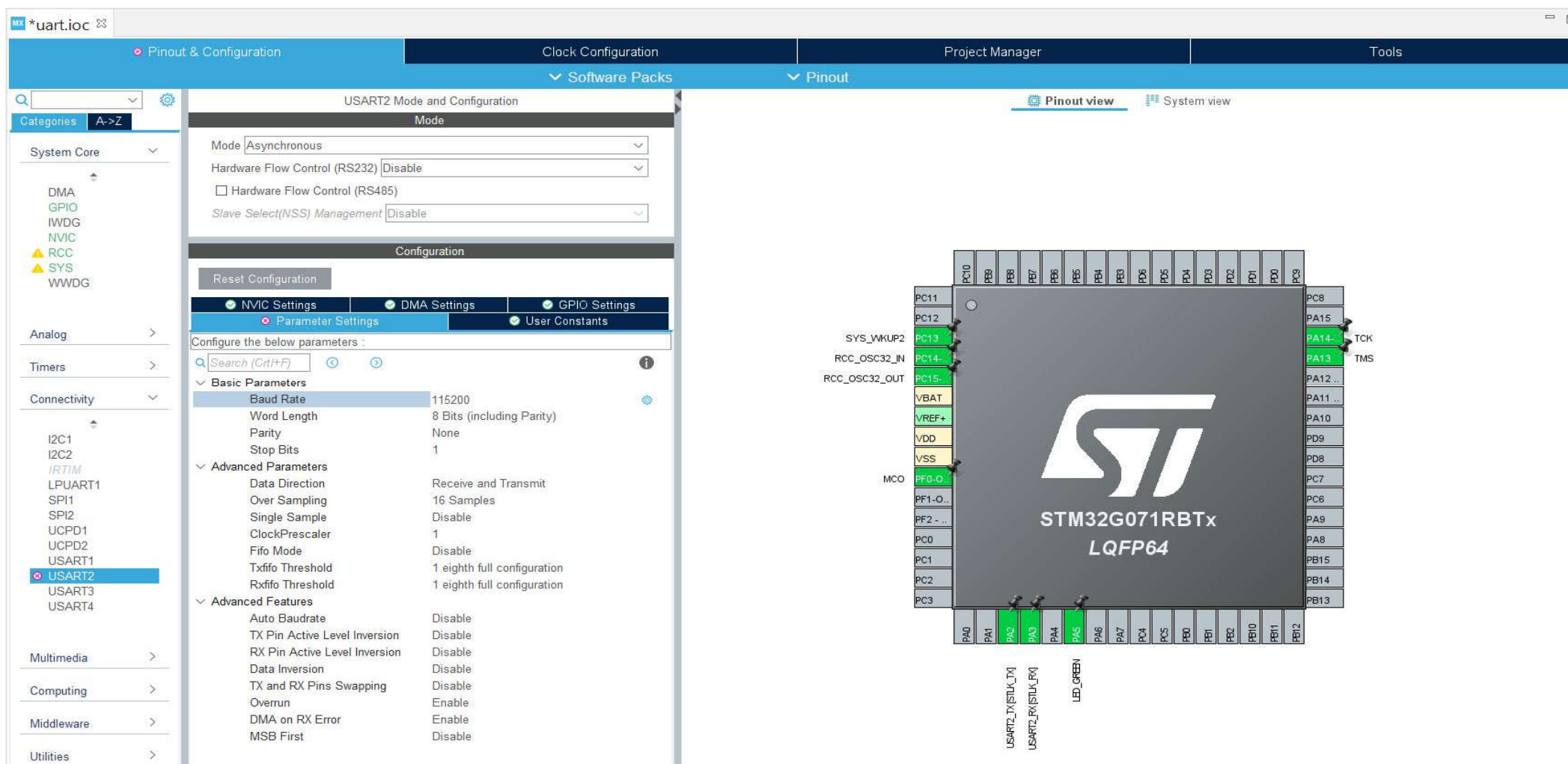
```
;uint8_t MSG[35] = {'\0'}  
;uint8_t X = 0  
;()HAL_Init  
;()SystemClock_Config  
;()MX_GPIO_Init  
;()MX_USART2_UART_Init
```

# التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط الـ Polling لطباعة قيمة متحول على النافذة التسلسلية

```
while (1)  
}  
printf(MSG, "Hello Dudes! Tracing X =  
;%d\r\n", X)  
HAL_UART_Transmit(&huart2, MSG,  
sizeof(MSG), 100)  
HAL_Delay(500)  
;++X  
{{
```

# التطبيق العملي 2 : استخدام المنفذ التسلسلي UART2 في نمط ال interrupt لطباعة قيمة متحول على النافذة التسلسلية

نقوم بضبط إعدادات المنفذ التسلسلي: 



The screenshot displays the STM32CubeMX Pinout & Configuration window for the USART2 peripheral. The left sidebar shows the 'System Core' and 'Connectivity' sections, with 'USART2' selected under 'Connectivity'. The main window is divided into 'Mode' and 'Configuration' tabs. The 'Mode' tab shows 'Asynchronous' mode, 'Disable' for Hardware Flow Control (RS232), and 'Disable' for Slave Select(NSS) Management. The 'Configuration' tab shows 'Reset Configuration' and 'Parameter Settings' selected. The 'Parameter Settings' section shows the following parameters:

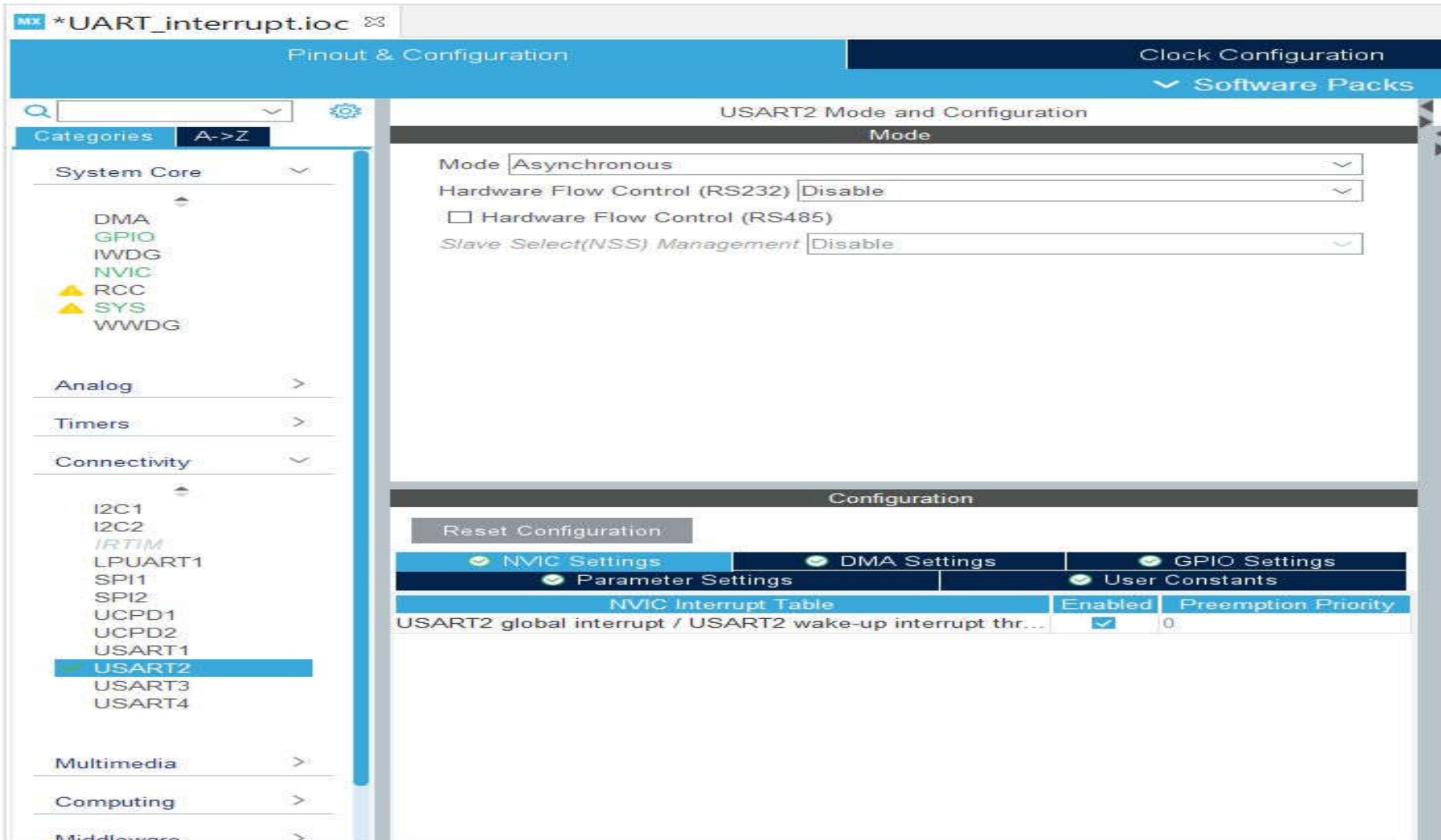
- Baud Rate: 115200
- Word Length: 8 Bits (including Parity)
- Parity: None
- Stop Bits: 1
- Advanced Parameters:
  - Data Direction: Receive and Transmit
  - Over Sampling: 16 Samples
  - Single Sample: Disable
  - ClockPrescaler: 1
  - Fifo Mode: Disable
  - Txfifo Threshold: 1 eighth full configuration
  - Rxfifo Threshold: 1 eighth full configuration
- Advanced Features:
  - Auto Baudrate: Disable
  - TX Pin Active Level Inversion: Disable
  - RX Pin Active Level Inversion: Disable
  - Data Inversion: Disable
  - TX and RX Pins Swapping: Disable
  - Overrun: Enable
  - DMA on RX Error: Enable
  - MSB First: Disable

The right sidebar shows the 'Pinout view' and 'System view' tabs. The 'Pinout view' shows the pin connections for the USART2 peripheral, including PC10, PC11, PC12, PC13, PC14, PC15, VBAT, VREF+, VDD, VSS, MCO, PF0-O, PF1-O, PF2-O, PC0, PC1, PC2, PC3, PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7, PA8, PA9, PA10, PA11, PA12, PA13, PA14, PA15, TCK, TMS, and LED\_GREEN. The pin connections for USART2 are shown as PA2 (TX) and PA3 (RX).



# التطبيق العملي 2 : استخدام المنفذ التسلسلي UART2 في نمط ال interrupt لطباعة قيمة متحول على النافذة التسلسلية

نقوم بضبط إعدادات مقاطعة المنفذ التسلسلي: 



The screenshot shows the STM32CubeMX Pinout & Configuration window. The left sidebar lists various system components, with **USART2** selected under the **Connectivity** category. The main panel displays the **USART2 Mode and Configuration** settings.

**USART2 Mode and Configuration**

**Mode**

- Mode: Asynchronous
- Hardware Flow Control (RS232): Disable
- ☐ Hardware Flow Control (RS485)
- Slave Select(NSS) Management: Disable

**Configuration**

Reset Configuration

NVIC Settings		DMA Settings		GPIO Settings	
Parameter Settings				User Constants	
NVIC Interrupt Table		Enabled	Preemption Priority		
USART2 global interrupt / USART2 wake-up interrupt thr...		<input checked="" type="checkbox"/>	0		

# التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط ال interrupt لطباعة قيمة متحول على النافذة التسلسلية

يصبح الكود بالشكل التالي: 

```
"include "main.h#  
;UART_HandleTypeDef huart2  
;uint8_t rx_buffer[4]  
;"uint8_t tx[]="hello  
;void SystemClock_Config(void)  
;static void MX_GPIO_Init(void)  
;static void MX_USART2_UART_Init(void)
```

# التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط ال- interrupt لطباعة قيمة متحول على النافذة التسلسلية

```
int main(void)  
{  
  
;()HAL_Init  
  
;()SystemClock_Config  
  
;()MX_GPIO_Init  
  
;()MX_USART2_UART_Init  
  
;HAL_UART_Transmit_IT(&huart2, tx, 5)  
  
;HAL_UART_Receive_IT(&huart2, rx_buffer, 4)
```

# التطبيق العملي 1 : استخدام المنفذ التسلسلي UART2 في نمط ال interrupt لطباعة قيمة متحول على النافذة التسلسلية

**while (1)**

**}**

**{ {**

**Void**

**HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef**  
**\*huart)**

**{**

**HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_5) ;**

**HAL\_UART\_Receive\_IT(&huart2, rx\_buffer, 4);**

**HAL\_UART\_Transmit\_IT(&huart2, rx\_buffer, 4);**

**}**

**Thank you for listening**