إدارة المقاطعات في متحكمات STM32

محتويات الجلسة:

1- Exceptions Overview
2- Micro-Coded Interrupts
3- Exception Types
4- System Exceptions
5- متحكم NVIC
6- أنماط عمل المعالج
7- مصطلح الـ Preemption
8- Interrupt Late Arrival
9- Interrupts Tail-Chaining
10- Interrupt Lifecycle
11- Reset Behavior
12- Exception Behavior
13- The Peripheral Pending bit
14- Configure Interrupts
15- STM32CubeMX Usage
16- STM32CubeHAL Usage

الأدوات اللازمة للجلسة:

- لوحة Nucleo-64bit
- كبل Type-A to Mini-B
- ليدات
- مفاتيح لحظية

1- **نظرة عامة عن مفهوم الاستثناءات:Exceptions Overview**

توفر معالجات ARM ميزة تـ سمى الا ستثناءات/المقاطعات. المقاطعة هي حدث غير متزامن يتـ سبب في إيقاف تنفيذ الكود الحالي اعتمادا على مبدأ الأولوية. تنـ شأ المقاطعات من خلال الـ hardware ـ أومن البرنامج نفسه الـ software ـ، ويمكن التحكم فيها عبر متحكم NVIC(Nested Vectored Interrupt Controller)، كما توفر معالجات ARM هيكلية خا صة للتعامل مع الا ستثناءات/ المقاطعات وتدعى Micro-Coded Architecture حيث يتم تكديس (تعليق) المقاطعات ثم البدء بمعالجتها حـ سب أولويتها ثم العودة لا ستكمال تنفيذ البرنامج الرئيـ سي وكل ذلك يتم Hardware أي بدون التدخل من قبل المعالج مما يعني سرعة ا ستجابة أعلى للا ستثناءات وتخفيف الـ ضغط عن المعالج بالإ ضافة إلى مرونة أعلى في العمل والقدرة على دعم أنظمة الزمن الحقيقي RTOS.

2- **Micro-Coded Interrupts:**

يتم الدخول إلى المقاطعة والخروج منها عن طريق الـ Hardware بغرض تقليل زمن التأخير بالإضافة إلى تأمين سرعة استجابة عالية للمقاطعات والاستثناءات بشكل عام، حيث يقوم العتاد الصلب (Hardware) بـ:

1. حفظ وا ستعادة processor context (وتـ شمل حالة المعالج لحظة حدوث المقاطعة بما في ذلك حالة وقيم المـ سجلات) عند حدوث المقاطعة وعند العودة منها لا ستكمال تنفيذ البرنامج الرئيـ سي انطلاقاً من التعليمة التي تم التوقف عندها عند حدوث المقاطعة.
2. يسمح باكتشاف الوصول المتأخر للمقاطعات/ الاستثناءات وخدمتها بناءً على مستوى الأولوية لها.
3. تـ سمح بخدمة المقاطعة المعلقة التالية عند الانتهاء من خدمة المقاطعة الحالية دون الحاجة لإعادة مرحلة (حفظ/ استعادة) حالة المعالج (processor context) مما يزيد من سرعة الاستجابة للمقاطعات وتدعي هذه الخاصية بـ tail-chaining.

3- **أنواع الاستثناءات:Exception Types**

هناك نوعين من الاستثناءات هي:

system exception : يتم توليدها من قبل المعالج نفسه داخلياً

interrupts : المقاطعات وتأتي من العالم الخارجي للمعالج

هناك 15 ا ستثناء من نوع system exception تدعمها معالجات CORTEX-M بالإ ضافة إلى 240 مقاطعة ، لذا بالمجمل فإن معالجات CORTEX-M تدعم 255 استثناء، بحيث:

1. الاستثناء رقم واحد هو للـ RESET
2. فقط 9 ا ستثناءات من نوع system exception يتم التعامل معها والـ 6 الباقية هي للا ستخدامات المستقبلية.
3. الاستثناء رقم 16 هو للمقاطعة رقم 1(IRQ1).

4- System Exceptions :

1. **RESET:** هو ا ستثناء من نوع system exception ، يتم طلب هذا الا ستثناء عند تغذية المعالج أو من خلال الضغط على زر الـ Reset الموجود على اللوحة التطويرية، حيث يقوم المعالج بالتوقف عن تنفيذ الأوامر والتوجه إلى العنوان الموجود ضمن ا ستثناء الـ Reset ضمن جدول أ شعة المقاطعة Vector table.
2. **Non-Maskable Interrupt (NMI)** :من ا سمها هي الاستثناءات التي لا يمكن إلغاء تفعيلها، يتم قدح هذا الا ستثناء عند حدوث خطأ ما عند تنفيذ أحد Exception Handler ، ولها أعلى أولوية بعد ا ستثناء الـ Reset، في متحكمات STM32 يتم ربط ا ستثناءات الـ NMI مع نظام حماية الـ ساعة Clock security system(CSS) ، حيث الـ CSS هي وحدة طرفية تقوم بفحص حالة الـ ساعة الخارجية HSE وفي حال اكتشاف مشكلة ما فيها تقوم بتعطيل HSE وتفعيل الساعة الداخلية HSI.
3. **Hard Fault**: يتم توليد هذا الاستثناء عند حدوث استثناء ما وفي حال عدم وجود آلية للتعامل مع هذا الا ستثناء، على سبيل المثال عند حدوث ا ستثناء معين وفي حال عدم تفعيل Exception Handler الخاص به يتم في هذه الحالة توليد استثناء الـ Hard Fault.
4. **Memory management fault:** يحدث هذا الا ستثناء عند محاولة الو صول لموقع غير مـ صرح بالو صول له في الذاكرة أو عندما يتم انتهاك قاعدة من قواعد حماية الذاكرة أثناء تنفيذ التعليمات البرمجية.
5. **BUS Fault:** يحدث هذا الا ستثناء عند حدوث خطأ أثناء الو صول إلى ذاكرة المعطيات أو التعليمات وقد يكون بسبب حدوث خطأ على الناقل Bus في النظام الذاكري.
6. **Usage Fault:** يحدث هذا الاستثناء أثناء تنفيذ التعليمات وقد يكون بسبب استخدام تعليمة غير موجودة أو عند محاولة الو صول لموقع غير مـ صرح به أو بـ سبب خطأ في الحالة الناتجة عن تنفيذ التعليمة مثلاً الحصول على قيمة غير صالحة مثل القسمة على صفر.
7. **SVC Call(System Service Call):** بتم توليد هذا الاستثناء عند استدعاء تعليمة Supervisor Call وهذه التعليمة تستخدم من قبل نظام الزمن الحقيقي Real Time Operation (RTC).
8. **Debug Monitor:** يتم توليد هذا الا ستثناء عند ا ستخدام ميزة اكت شاف الأخطاء البرمجية وعندها يكون المعالج في نمط Monitor Debug Mode .

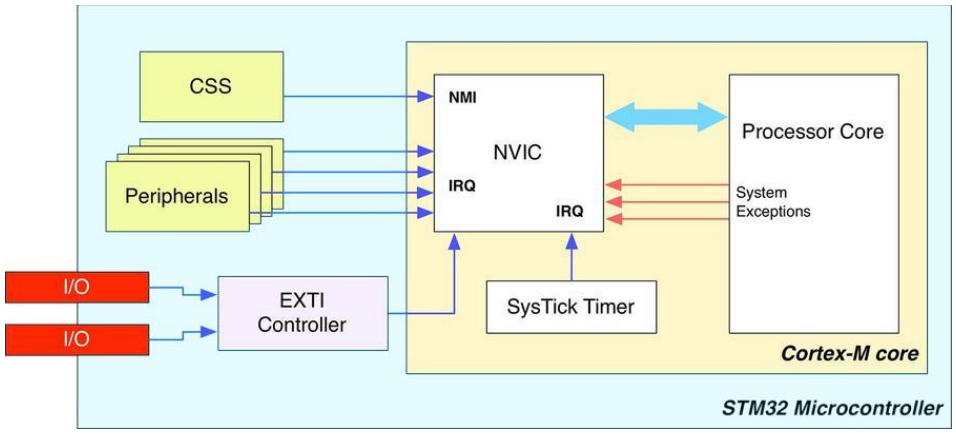
9. **PendSV:** وهو استثناء آخر يتعلق بالـ RTOS.
10. **System tick:** هذا الا ستثناء يـ ستخدم دائماً في أنظمة الزمن الحقيقي RTOS، حيث يحتاج كل نظام RTOS إلى مؤقت لجدولة أنشطة النظام بحيث يقوم بتوليد مقاطعة في كل فترة زمنية لترك تنفيذ المهمة الحالية وتنفيذ مهمة أخرى ويكون هذا المؤقت متصل داخلياً بنواة CORTEX-M وليس وحدة خارجية كباقي الطرفيات.
11. **Interrupt (IRQ):** هي عبارة عن الا ستثناءات التي يتم توليد ها من قبل الطرفيات أو من قبل البرنامج، حيث تستخدم الطرفيات المقاطعات بهدف الاتصال بالمعالج.

5- متحكم NVIC (Nested Vectored Interrupt Controller):

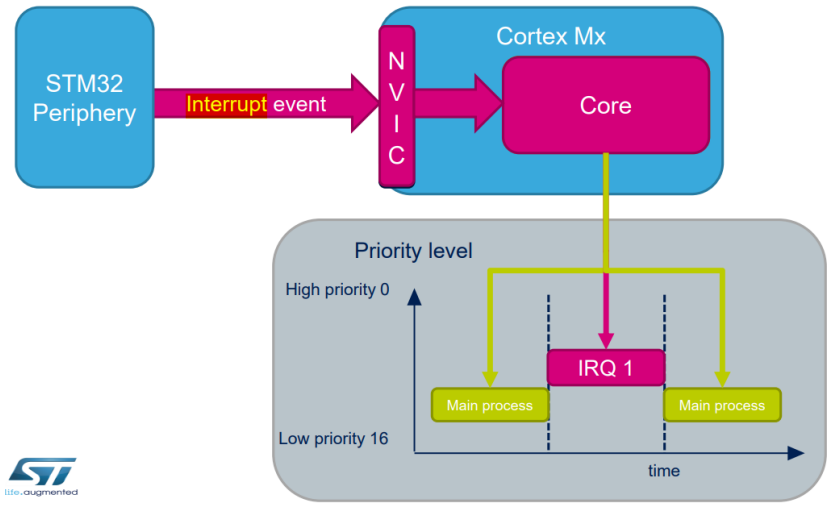
يعتبر Nested Vectored Interrupt Controller (NVIC) هو المتحكم الم سؤول عن تحديد أولو يات المقاطعات ، وتح سين أداء المعالج MCU وتقليل زمن ا ستجابة المقاطعة. يوفر NVIC أي ضًا خطوات محددة للتعامل مع المقاطعات التي تحدث أثناء تنفيذ مقاطعات أخرى أو عندما يكون المعالج في مرحلة ا ستعادة حالته السابقة واستئناف عمليته المعلقة التي توقف عندها بسبب حدوث المقاطعة.

حيث :

1. يتم ربط مقاطعة Clock Security System (CSS) مع خطوط مقاطعات الـ Non-Maskable Interrupt (NMI) وهي المقاطعات التي لا يمكن تجاهلها أو إلغاء تفعيلها.
2. يتم ربط مقاطعات الطرفيات (مثل مقاطعات المؤقتات، المبدلات التشابهية رقمية وغيرها من الطرفيات) مع خطوط Interrupt Requests (IRQ)
3. المقاطعات الخارجية القادمة من الـ GPIO يتم ربطها مع External Interrupt/Event Controller (EXTI) ليتم بعد ذلك ربطها مع خطوط IRQ ، كما هو موضح بالشكل التالي:

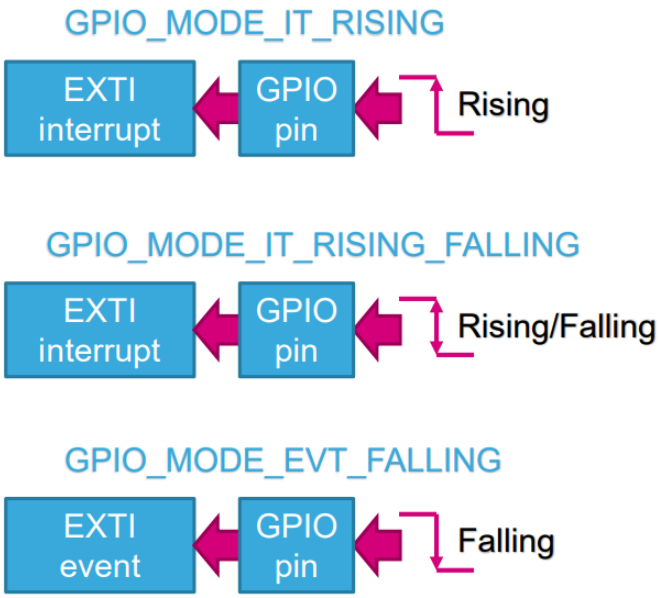


الشكل (1): *NVIC module in STM32 MCUs*



الشكل (2):دور متحكم NVIC

يتم تجميع المقاطعات الخارجية ضمن خطوط متصلة مع أطراف المتحكم المصغر GPIO، وبما أن للمعالج عدة أطراف GPIOS لذا فإن كل خط من خطوط المقاطعات الخارجية EXTI هو عبارة عن خط مشترك بين عدة أطراف Pins، في كل خط من خطوط المقاطعات الخارجية (كل مجموعة) يحق لـ pin واحد أن يقوم بتوليد المقاطعة وعلى البرنامج أن يكتشف أي pin قام بتوليد المقاطعة حيث يمكن قدح المقاطعة عند الجبهة الصاعدة rising edge أو الهابطة falling edge أو عند كليهما .



الشكل (3):GPIO_MODE_IT

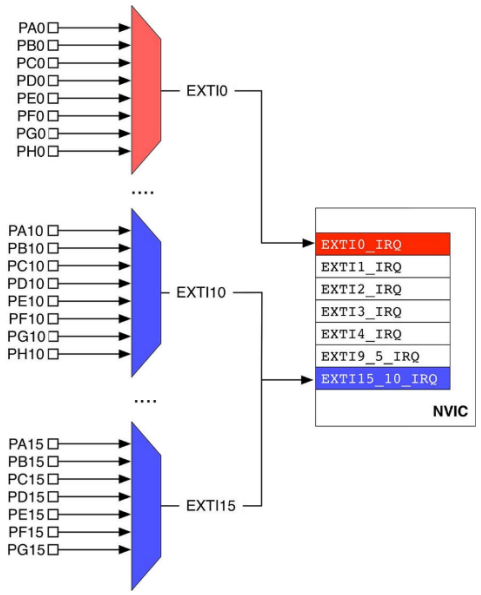
يحتوي متحكم STM32 على 16 خط للمقاطعات الخارجية هي من الخط 0 حتى الخط 15حيث تشير أرقام الخطوط إلى أرقام اطراف الـ GPIOS - ، هذا يعني أن الطرف PA0 متصل بالخط LINE0 والطرف PA13 متصل بالطرف LINE13 ، أيضاً PB0 و PC0 متصلين بالخط LINE0 ، بمعنى أن جميع الأطراف ذات الأرقام PX0

متصلة بالخط LINE0 (حيث يعبر x عن اسم المنفذ)، أيضاً جميع الأطراف ذات الأرقام PX3 متصلة بالخط LINE3 وهكذا...

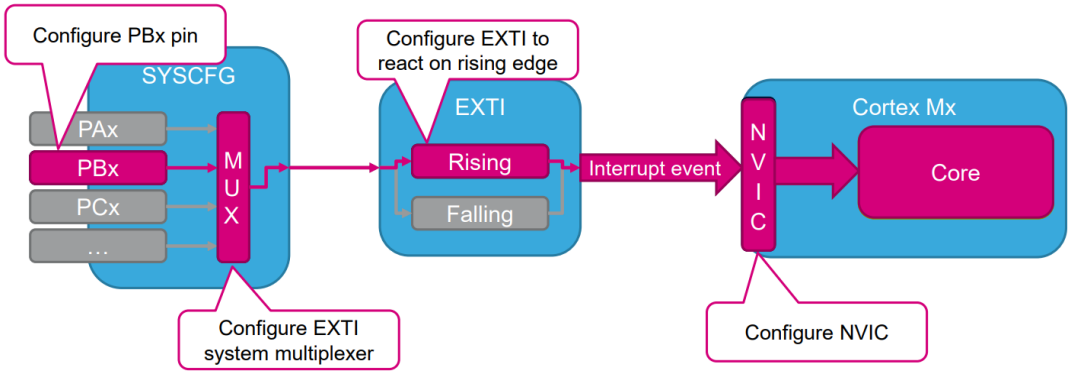
لكن يجب الانتباه للملاحظات التالية:

1. الأطراف PA0، PB0، PC0... جميعها متصلة بالخط LINE0 لذا في اللحظة الواحدة بإمكانك توليد مقاطعة على طرف واحد فقط من هذه الأطراف.
2. الأطراف PA0، PA5 متصلين على خطين مختلفين من خطوط المقاطعة لذا يمكنك استخدامهم في نفس اللحظة لتوليد المقاطعة.

في حال حدوث المقاطعة فإن المعالج يتوقف عن تنفيذ الكود الحالي ويقوم بمعالجة المقاطعة من خلال تنفيذ برنامج خدمة المقاطعة (Interrupt Service Routines (ISR)) والذي يتم تحديد عنوانه في الذاكرة من خلال جدول أشعة المقاطعة المعرف مسبقاً (Vector Interrupt Table (VIC)).



الشكل (4): External Interrupt lines



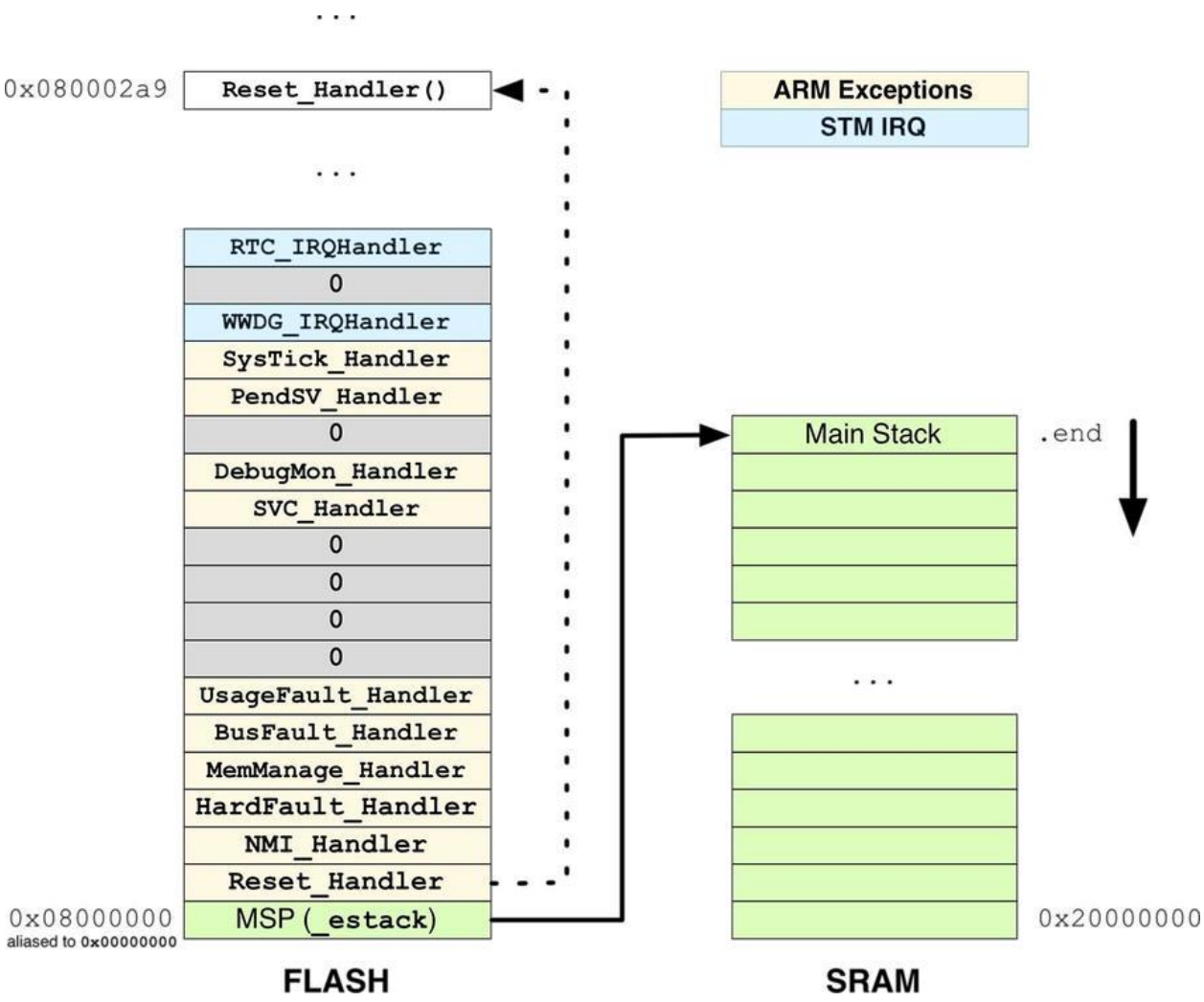
الشكل (5): Interrupt Mechanism

Number	Exception Type	Priority	Function
1	Reset	-3	Reset
2	NMI	-2	Non-Maskable Interrupt
3	Hard Fault	-1	All faults that hang the processor
4	Memory Fault	Configurable	Memory issue
5	Bus Fault	Configurable	Data bus issue
6	Usage Fault	Configurable	Data bus issue
7 ~ 10	Reserved	—	Reserved
11	SVCall	Configurable	System service call (SVC instruction)
12	Debug	Configurable	Debug monitor (via SWD)
13	Reserved	—	Reserved
14	PendSV	Configurable	Pending request for System Service call
15	SysTick	Configurable	System Timer
16 ~ 240	IRQ	Configurable	Interrupt Request

جدول أشعة المقاطعة Vector interrupt table

حيث يتم تعريف جدول أ شعة المقاطعة Vector interrupt table ضمن كود مكتوب بلغة الأ سمبلي ضمن ملف الإقلاع للمعالج startup_*.s .

حسب الاصطلاح ، يبدأ جدول Vector interrupt table عند العنوان 0x00000000 في جميع المعالجات المبنية اعتماداً على Cortex-M. إذا كان جدول Vector interrupt table موجودًا في ذاكرة الفلاش الداخلية (و هذا ما يحدث عادةً) ، وبما أن ذاكرة الفلاش في جميع متحكمات STM32 تبدأ من العنوان0x08000000 ،لذا يتم و ضع جدول Vector interrupt table بدءًا من العنوان 0x08000000 ، ويتم ت سميته بـ 0x00000000- عند إقلاع المعالج.



الشكل (6):Vector Interrupt Table in ARM cores

حيث يـ شير العنوان 0x00000000 إلى عنوان مؤ شر المكدس Main Stack Pointer (MSP) ضمن الذاكرة SRAM

6- أنماط عمل المعالج Processor mode

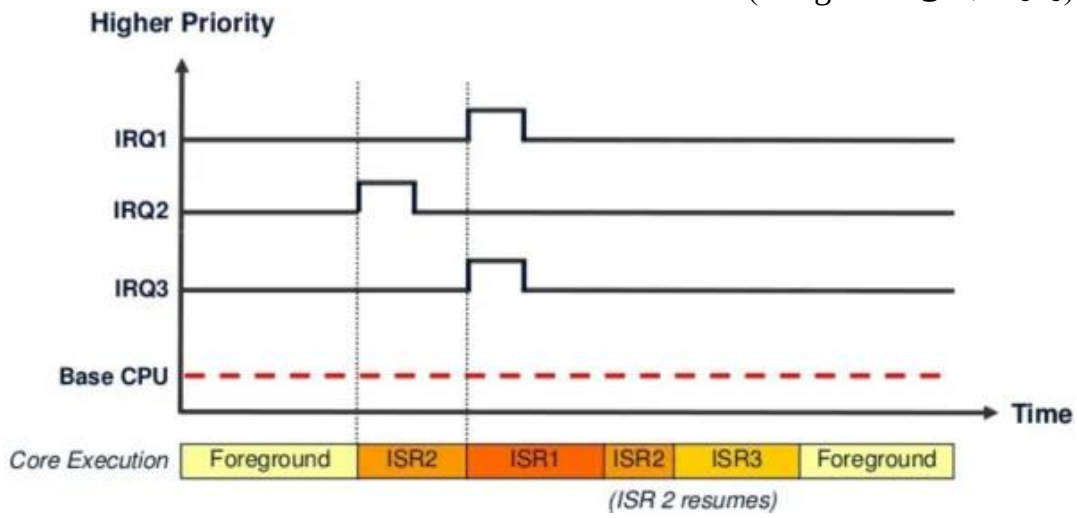
للمعالج نمطي عمل أساسين:

Thread Mode: يكون المعالج في نمط Thread Mode في الو ضع الطبيعي له عند تنفيذ الكود أو عند عمل reset له .

Handler Mode: يدخل المعالج في نمط Handler Mode عند حدوث ا ستثناء /مقاطعة حيث يتم حفظ المكان الذي تم توقف المعالج عنده كما يتم حفظ حالة الم سجلات والأعلام flags بـ شكل آلي وعن طريق الـ hardware - ليضمن استجابة سريعة لحدث المقاطعة.

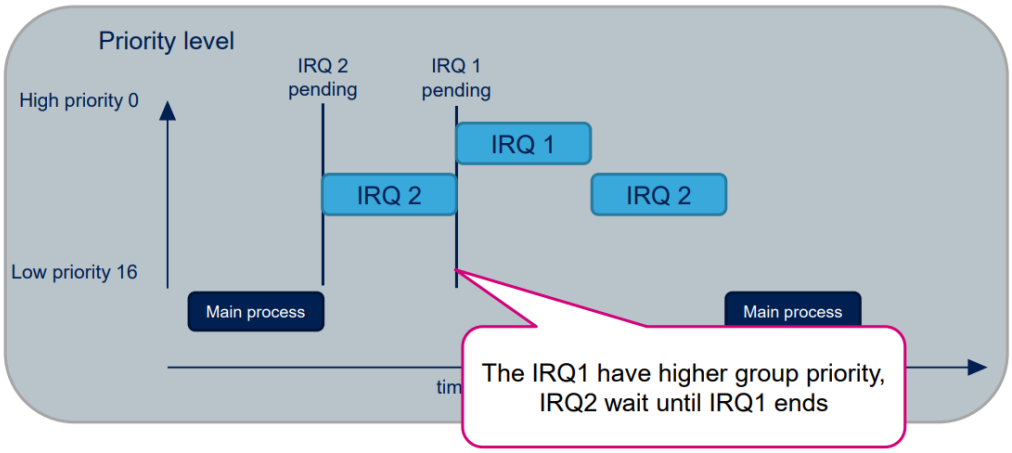
7- Preemption:

مـ صطلح Preemption يعني مقاطعة عمل task أو مهمة معينة بـ سبب حدوث task أو مهمة أخرى لها أولوية أعلى في التنفيذ وفي هذه الحالة فإن الـ task التي تمت مقاطعتها تسمى preempted.
فعند حدوث عدة مقاطعات أو استثناءات يتم تعليقها جميعاً وتنفيذ الاستثناء أو المقاطعة ذات الرقم الأصغر في جدول أ شعة المقاطعات VIT الذي تحدثنا عنه سابقاً، وأثناء تنفيذ هذه الا ستثناء / المقاطعة من قبل المعالج وفي حال حدوث ا ستثناء/ مقاطعة ذات أولوية أعلى يتم عمل Preemption للمقاطعة الحالية والذهاب لتنفيذ المقاطعة ذات الأولوية الأعلى، حيث المقاطعة ذات الرقم الأ صغر في جدول أ شعة المقاطعات لها أولوية أعلى في التنفيذ من قبل المعالج.
على سبيل المثال:
بفرض حدوث ثلاث ا ستثناءات/مقاطعات لها مـ ستويات أولوية مختلفة، وبفرض أن الا ستثناء IRQ1 قاطع عمل الا ستثناء IRQ2 (preemption) أثناء تنفيذه وقام بتعليق الا ستثناء IRQ3 (pend) لحين انتهاء IRQ1، فعند الانتهاء من تنفيذ برنامج خدمة المقاطعة ISR الخاص بالا ستثناء IRQ1 ، يتم إكمال تنفيذ برنامج خدمة المقاطعة الخاص بالا ستثناء IRQ2 من مكان توقفه عند مقاطعته من قبل الا ستثناء IRQ1 وعند إكمال تنفيذه يتم تنفيذ الا ستثناء المعلق IRQ3 وعند الانتهاء من تنفيذ برنامج خدمة المقاطعة ISR3 الخا صة به يتم ا ستعادة العنوان الذي توقف المعالج عنده عند حدوث الا ستثناءات لي ستكمل تنفيذ التعليمات (وهو ما يدعى foreground).

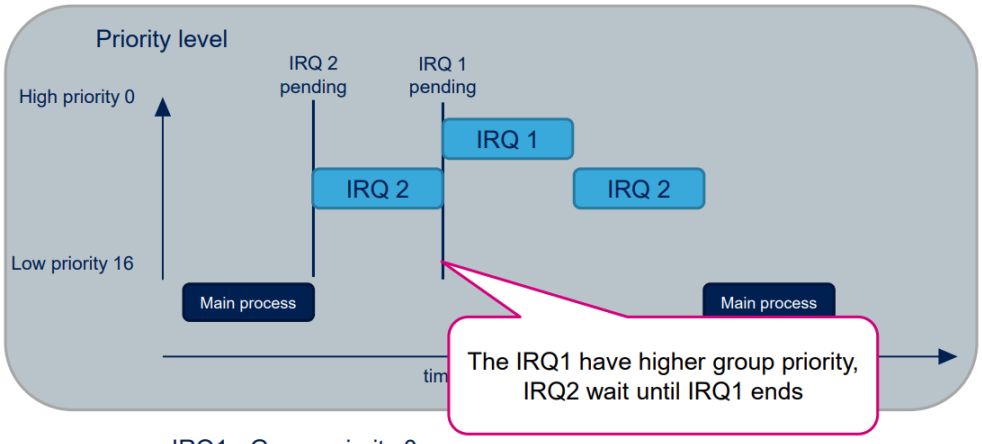


الشكل (7):Preemption allow IRQ1 to be executed

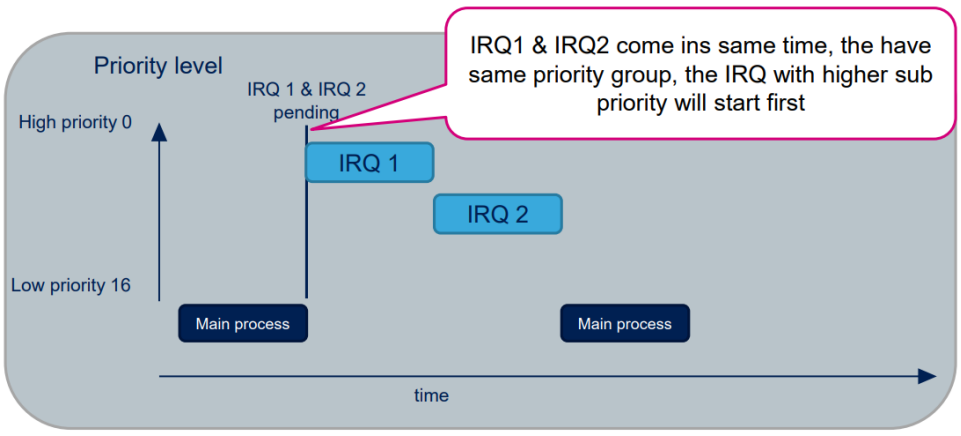
أصبح لدينا مصطلحين :
Preemption: ويعني مقاطعة استثناء يتم حالياً تنفيذه من قبل المعالج عند حدوث استثناء ذو مستوى أولوية أعلى.
Pending: ويعني تعليق ا ستثناء حدث ولكن لم يتم البدء بتنفيذه ، لحين تنفيذ الا ستثناءات ذات الأولوية الأعلى.



- IRQ1 - Group priority 0
- IRQ2 - Group priority 1



- IRQ1 - Group priority 0
- IRQ2 - Group priority 1



- IRQ1 - Group priority 0, Sub priority 0
- IRQ2 - Group priority 0, Sub priority 1

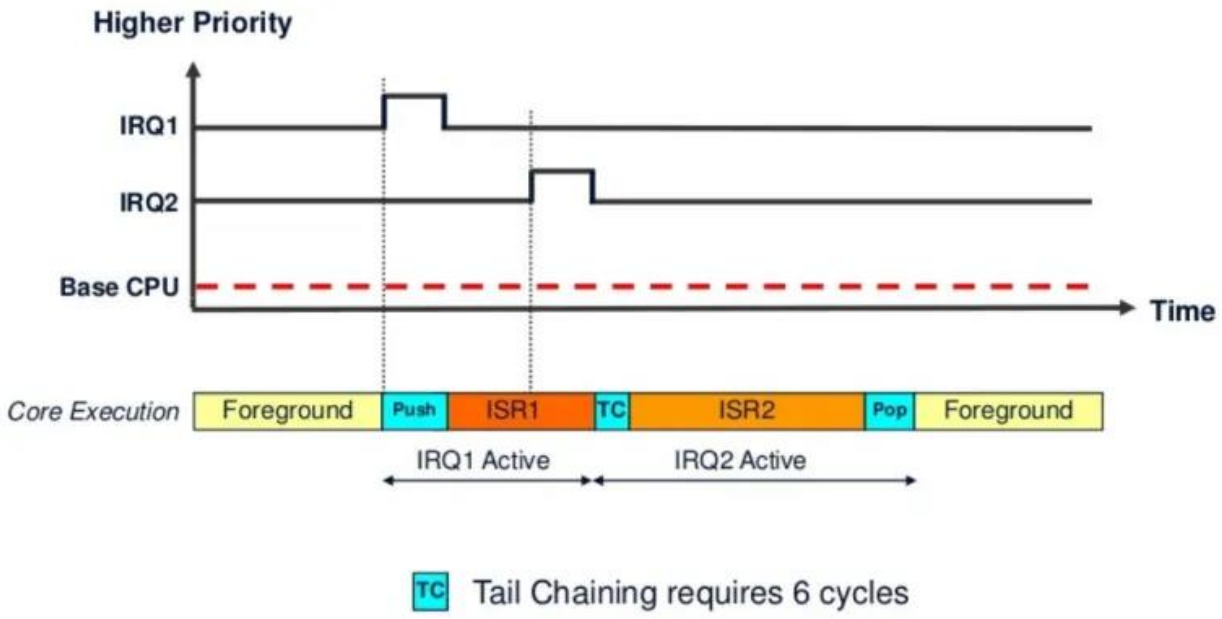
الشكل (8): الحالات المختلفة لحدوث المقاطعات

8- Interrupts Tail-Chaining:

عند حدوث مقاطعة، يتم حفظ محتوى الكود الحالي الذي يقوم المعالج حالياً بتنفيذه إلى المكدس وتسمى هذه العملية pushed، ويتوجه المعالج لجدول أشعة المقاطعة لمعرفة عنوان برنامج خدمة المقاطعة ISR التي حدثت ومن ثم يبدأ بتنفيذه، وفي نهاية برنامج خدمة المقاطعة ISR يتم استعادة محتوى الكود الذي كان المعالج يقوم بتنفيذه من المكدس وتسمى هذه العملية popped out ويقوم المعالج باستكمال تنفيذ الكود وتسمع هذه العملية foreground .

لكن في حال كان هناك استثناء/مقاطعة جديدة معلقة ، في هذه الحالة وعند الانتهاء من تنفيذ برنامج خدمة المقاطعة ISR يتم تجاهل عمليتي push/pop وينتقل المعالج لتنفيذ برنامج خدمة المقاطعة التالية دون الحاجة للقيام بإجراءات إضافية وهو ما يسمى بـ *Tail-Chaining*. - وهو مايحتاج فقط لـ - 6 cycles في معالجات Cortex-M3/M4 وهو ما يعني سرعة عالية جداً في الأداء وفي زمن الاستجابة للمقاطعات بالتالي يقلل زمن انتظار المقاطعات (interrupt latency).

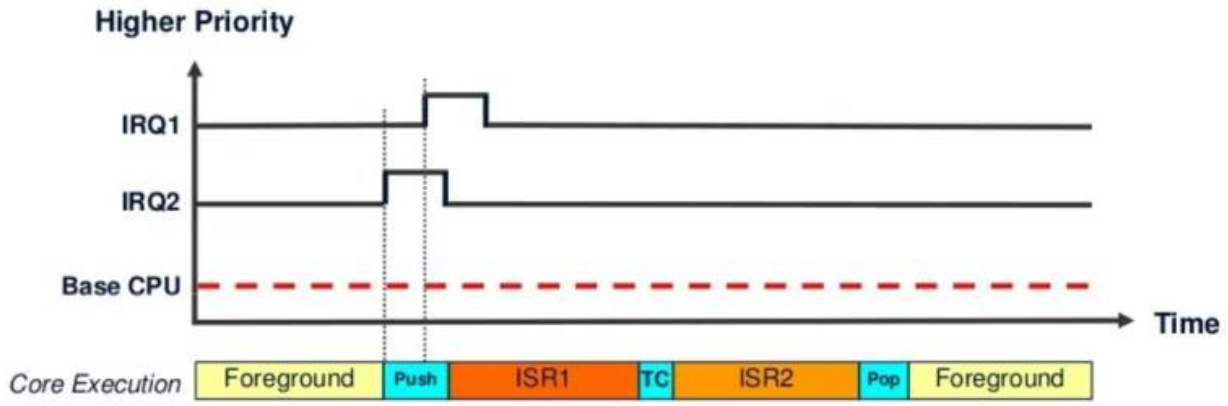
يوضح الشكل التالي حالة حدوث مقاطعة IRQ2 أثناء تنفيذ المعالج لبرنامج خدمة المقاطعة IRQ1.



الشكل (8):Tail chaining when IRQ2 comes while IRQ1 is executing

9- تأخر وصول المقاطعة Interrupt Late Arrival

بإمكان معالجات ARM اكتشاف الاستثناءات ذات الأولوية الأعلى حتى في مرحلة البدء بالاستجابة لاستثناء ما (exception entry phase) أي في مرحلة حفظ حالة المعالج والمسجلات والبحث عن عنوان برنامج خدمة المقاطعة في جدول أشعة المقاطعة، وفي هذه الحالة يتم التوجه إلى الاستثناء ذو الأولوية الأعلى وتعليق تنفيذ الاستثناء الحالي مع عدم الحاجة لإعادة مرحلة حفظ حالة المعالج والمسجلات، وهذه الخاصية تدعى Interrupt Late Arrival، ثم بعد الانتهاء من تنفيذ هذا الاستثناء الذي وصل متأخراً يتم الرجوع إلى الاستثناء الذي تم تعليقه وتنفيذ برنامج خدمة المقاطعة الخاصة به دون الحاجة لاستعادة حالة المعالج وهذا ما يدعى Tail chaining.



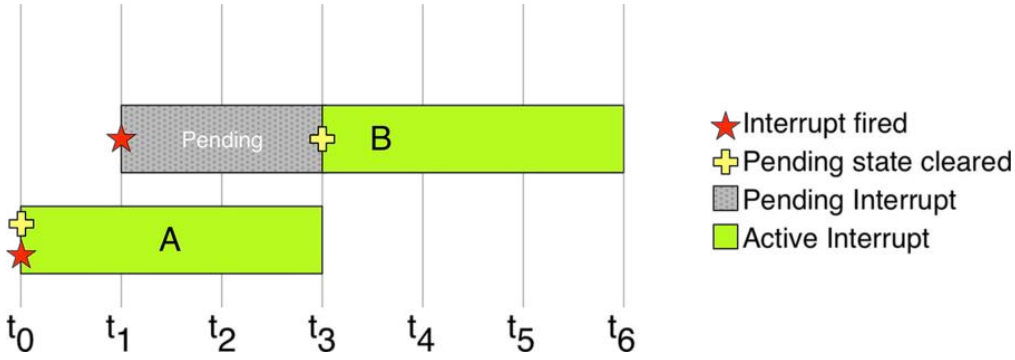
الشكل (9): Late arrival is detected when IRQ1 comes while IRQ2 is about to start

10-دورة حياة المقاطعة Interrupt Lifecycle

يمكن للمقاطعة أن تكون:

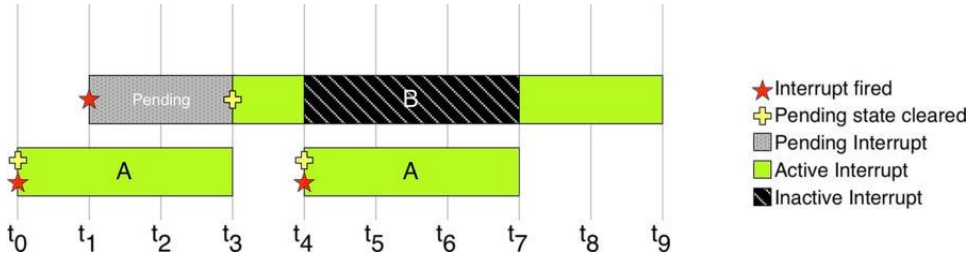
- مفعلة enabled أو غير مفعلة disabled (الوضع الافتراضي)
- معلقة (على قائمة الانتظار) Pending أو غير معلقة not pending
- فعالة active (يتم تنفيذها حالياً) أو غير فعالة not active.

عند حدوث مقاطعة يتم تعليقها لحين يـ صبح المعالج قادر على خدمتها وفي حال عدم وجود مقاطعات أخرى يتم حالياً تنفيذها، يتم تصفير حالة التعليق pending state تلقائياً من قبل المعالج ومن ثم يبدأ بخدمتها.



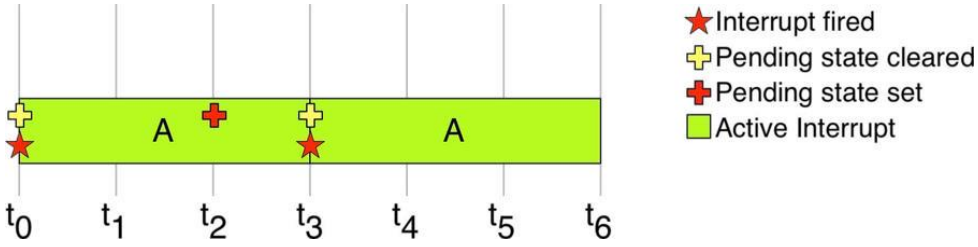
الشكل (10): ISR A then ISR B

المقاطعات ذات الأولوية الأدنى عليها الانتظار لحين عدم وجود مقاطعات بأولوية أعلى منها ، وعند البدء بتنفيذها يتم وضعها في حالة inactive (preempted)عند حدوث مقاطعة لها أولوية أعلى .

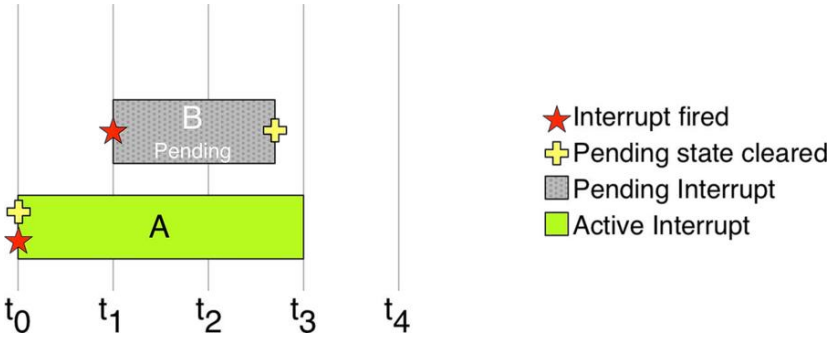


الشكل (11): ISR A preempts ISR B

يمكن قدح المقاطعة مجدداً أثناء تنفيذها من خلال تفعيل بت الانتظار pending bit الخاص بها ، كما يمكن إلغاؤها من خلال مسح بت الانتظار الخاص بها كما هو بالأشكال التالية.



الشكل (12):ISR A then ISR A



الشكل (13):ISR B canceled

Reset Behavior-11

عند حدوث مقاطعة التصفير Reset :

1. يتم تحميل القيمة الافترا ضية لمؤ شر المكدس الرئي سي (main stack pointer) MSP من العنوان0x00000000 والذي يحتوي على العنوان الأخير في الذاكرة RAM _estack
2. يتم تحميل عنوان الـ reset handler من العنوان 0x00000004
3. يتم تنفيذ الـ reset handler في نمط الـ thread mode
4. يعود المعالج لينفذ البرنامج الرئيسي

Exception Behavior-12:

عند حدوث استثناء ما، يتم إيقاف تنفيذ التعليمات الحالية ويتم توجيه المعالج لجدول أشعة المقاطعة حيث:

1. يتم تحميل عنوان exception handler الخاص بالاستثناء الحاصل من جدول أشعة المقاطعة
2. يبدأ المعالج بتنفيذ exception handler ويكون المعالج في هذه الحالة في نمط handler mode
3. يعود المعالج لإكمال تنفيذ التعليمات التي توقف عندها بسبب حدوث الاستثناء.

لشرح ما سبق بالتفصيل:

1. Interrupt Stacking (Context Saving):

1- يقوم المعالج بإنهاء التعليمة الحالية التي يقوم بتنفيذها طالما أنها لا تحتاج لأكثر من دورة تعليمة
2- يتم حفظ حالة المعالج (عنوان التعليمة التي وصل لعندها حالياً) إلى المكدس حيث يتم دفع(Pushed) ثمانية مسجلات (PC, R0-R3, R12, LR, xPSR) إلى المكدس.
3- يتم تحميل عنوان برنامج خدمة المقاطعة الحاصلة من جدول أشعة المقاطعة.

4- يتم البدء بتنفيذ برنامج خدمة المقاطعة ISR ، ويـ ستغرق تنفيذ كامل هذه العملية 12 cycles في معالجات M3/M4

2. برنامج خدمة المقاطعة Interrupt Service Routine (ISR) Handling:

1- لابد في البداية أن يقوم برنامج خدمة المقاطعة بتصفير علم المقاطعة التي قامت باستدعائه

2- بالنظر إلى أن بعض الا ستثناءات / المقاطعات يجب أن يتم خدمتها مئات أو آلاف المرات في الثانية. لذلك يجب أن يتم تنفيذها بـ سرعة كبيرة ولا يجب ا ستخدام تعليمات التأخيرالزمني (delay) ضمن برنامج خدمة المقاطعة ISR

3. العودة من برنامج خدمة المقاطعة (استعادة الحالة التي كان عندها المعالج قبل حدوث المقاطعة):

3- في نهاية برنامج خدمة المقاطعة يتم التأكد من عدم وجود مقاطعة في الانتظار(tail-chaining interrupt) من أجل الانتقال لتنفيذ برنامج خدمة المقاطعة التالية دون ا ستعادة حالة المعالج السابقة بهدف زيادة سرعة الاستجابة للمقاطعة.

4- يتم تنفيذ تعليمة EXC_RETURN لا ستعادة حالة المعالج قبل حدوث المقاطعة وا ستعادة حالة مسجلات المعالج (pop the CPU registers)

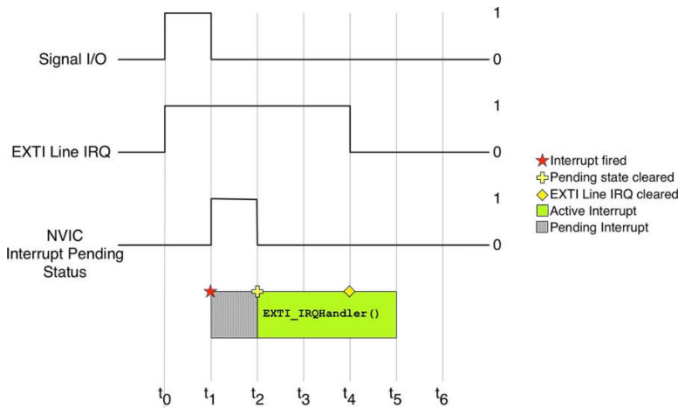
5- تـ ستغرق العودة من المقاطعة (ا ستعادة حالة المعالج)في معالجات ARM Cortex-M3/M4 حوالي 10 clock cycles

13-The Peripheral Pending bit:

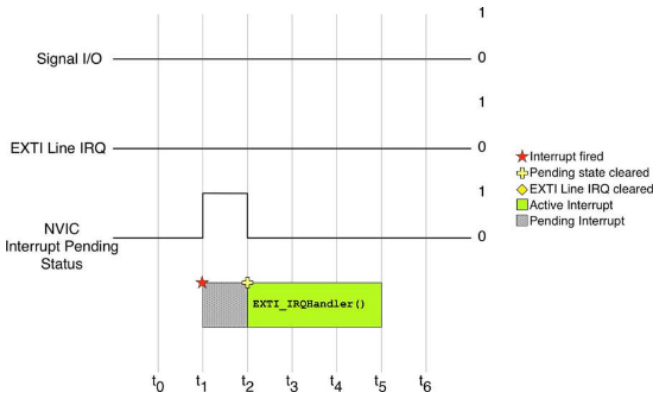
عند حدوث مقاطعة ، فإن معظم الأجهزة الطرفية لمتحكمات STM32 تقوم بإر سال إ شارة تأكيد لمتحكم الـ NVIC ، والتي يتم تعيينها في الذاكرة الطرفية من خلال بت مخ صص يدعى Peripheral Pending bit حيث يتم تهيئته بواحد منطقي ولايتم تصفيره آلياً وإنما يجب تصفيره يدوياً ضمن الكود.

يختلف بت ISR Pending bit عن بت Peripheral Pending bit، عندما يبدأ المعالج في تنفيذ برنامج خدمة المقاطعة ISR ، يتم مـ سح ISR Pending bit تلقائيًا ، ولكن سيتم الاحتفاظ بحالة Peripheral Pending bit لحين يتم تصفيره يدوياً ضمن الكود.

في حالة لم يتم تـ صفير بت Peripheral Pending ، فـ سيتم إعادة قدح المقاطعة مرة أخرى و سيتم تنفيذ برنامج خدمة المقاطعة ISR مرة أخرى.



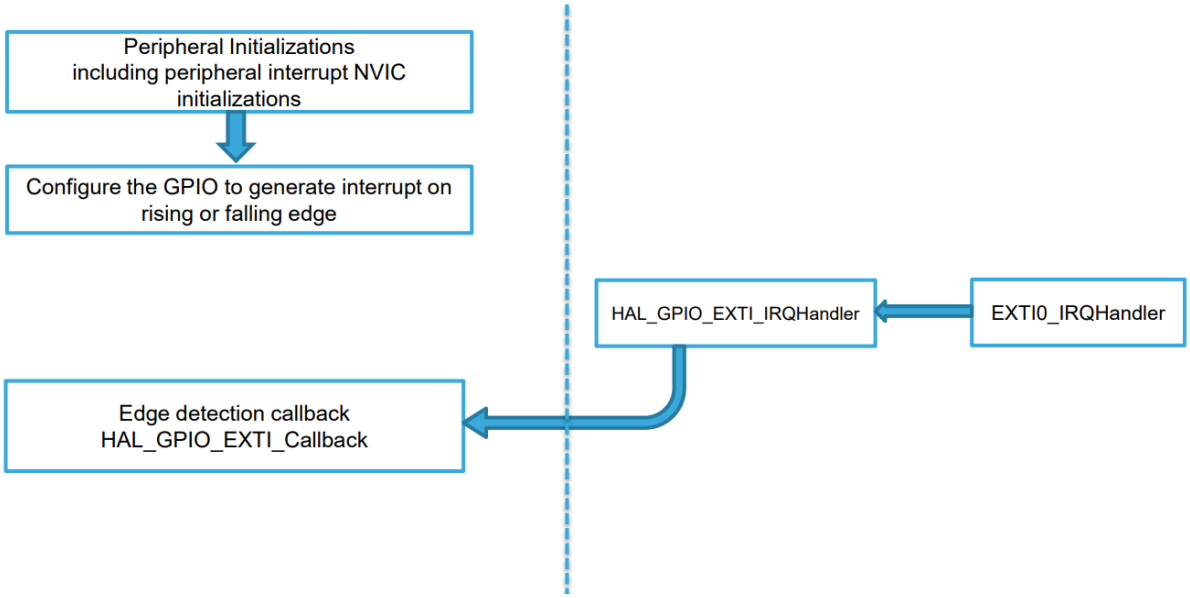
الشكل (14):External Peripheral Pending set



الشكل (15):Manually Peripheral Pending set

14-ضبط المقاطعات في متحكمات STM32 :

عند إقلاع المتحكم تكون فقط الا ستثناءات Reset, NMI and Hard Fault مفعلة بـشكل افترا ضي، بينما تكون باقي الاستثناءات والمقاطعات في حالة عدم تفعيل ويتم تفعيلها عند الطلب.



الشكل (16):آلية حدوث المقاطعة

لتهيئة أحد الطرفيات للعمل في وضع المقاطعة، يجب على كود المستخدم أن:

1. أن يقوم بتفعيل خط المقاطعة على سبيل المثال من أجل تفعيل المقاطعة على الطرف PA0 يجب تفعيل خط المقاطعة **EXTI0**، ضبط الجبهة (صاعدة/هابطة/ كليهما)
2. تنفيذ برنامج interrupt handler المعرف مسبقاً ضمن ملف الإقلاع ، مثلاً EXTI0_1_IRQHandler لمعالجة المقاطعة القادمة من الخط صفر EXTI0 والخط واحد EXTI1
3. داخل برنامج interrupt handler يجب:
   1- معرفة مصدر المقاطعة مثلاً الطرف PA0 أو PB0....
   2- تصفير علم المقاطعة

3- استدعاء برنامج خدمة المقاطعة ISR

على سبيل المثال الـ interrupt handler - الخاص بالمقاطعة الخارجية للقطب PC13 يكون بالشكل التالي:

```
void EXTI4_15_IRQHandler(void)
{

  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);

}
```

إذا ما قمنا بالـ ضغط بالزر الأيمن للفأرة على التابع HAL_GPIO_EXTI_IRQHandler واخترنا الخيار open declaration يظهر لنا ما بداخل هذا التابع بالشكل التالي:

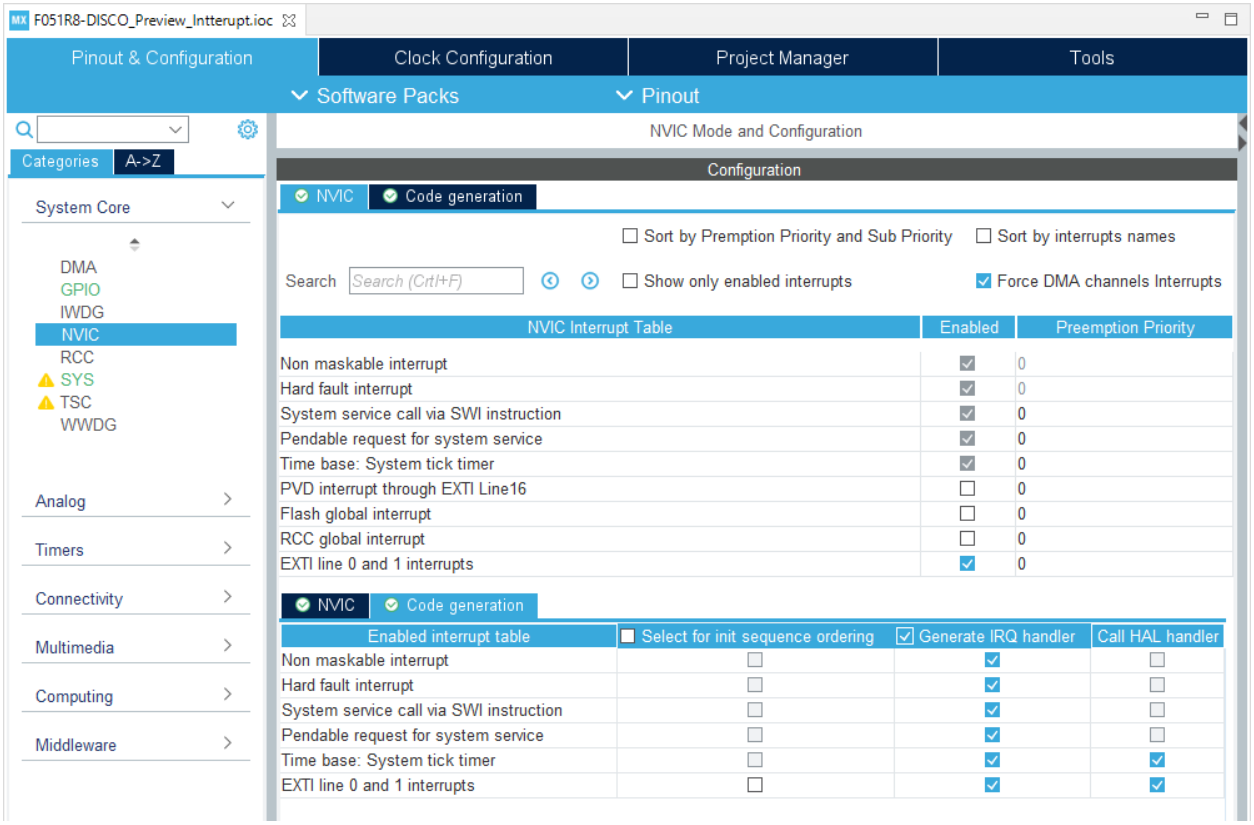
```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if (__HAL_GPIO_EXTI_GET_RISING_IT(GPIO_Pin) != 0x00u)
  {
    __HAL_GPIO_EXTI_CLEAR_RISING_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Rising_Callback(GPIO_Pin);
  }

  if (__HAL_GPIO_EXTI_GET_FALLING_IT(GPIO_Pin) != 0x00u)
  {
    __HAL_GPIO_EXTI_CLEAR_FALLING_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Falling_Callback(GPIO_Pin);
  }
}
```

حيث يتم في البداية فحص الجبهة التي حدثت عندها المقاطعة صاعدة/هابطة، ثم يتم تـ صفير علم المقاطعة وأخيراً يتم استدعاء برنامج خدمة المقاطعة HAL_GPIO_EXTI_FALLING_Callback، حيث يقوم المستخدم بكتابة الكود الذي يريد تنفيذه عند حدوث المقاطعة ضمنه.

15-ضبط الاستثناءات من خلال STM32CubeMx (أو من خلال STM32CubeIde) :

يمكن ضبط إعدادات المتحكم NVIC ضمن برنامج STM32CubeIde من خلال شريط Pinout & Configuration tab حيث تـ ضم هذه الـ شا شة قائمة بجميع المقاطعات الموجودة ضمن المتحكم وتؤمن خيارات ضبط هذه المقاطعات والتي تشمل:

1. تفعيل أو إلغاء تفعيل أي مقاطعة، بعض الا ستثناءات كما ذكرنا سابقاً تكون مفعلة تلقائياً، لاحظ أن برنامج STM32CubeIDE يجبرك على تفعيل الا ستثناء SysTick لاحتياج مكتبة Hal لاستخدامه.
2. ضبط Preemption Priority لكل مقاطعة
3. توليد IRQ handlers (in *_it.c) واستدعاء HAL callbacks والتي بدورها تقوم بتـ صفير pending bit واستدعاء برنامج خدمة المقاطعة الذي يقوم المستخدم بالتعديل عليه.



الشكل (17):NVIC configuration in STM32CubeMX/STM32CubeIDE

16- توابع مكتبة HAL المستخدمة مع المقاطعات:

لتفعيل مقاطعة IRQ فإن مكتبة HAL تؤمن التابع التالي:

void HAL_NVIC_EnableIRQ(IRQn_Type IRQn);

حيث IRQn_Type هو نوع البيانات المـ ستخدم مع جميع الا ستثناءات والمقاطعات لكل متحكم مـ صغر حيث يتم تعريفها والتصريح عنها ضمن ملف stm32fxxxx.h، على سبيل المثال من أجل المتحكمSTM32F030R8 يكون اسم الملف stm32f030x8.h

التابع المستخدم لإلغاء تفعيل مقاطعة IRQ:

void HAL_NVIC_DisableIRQ(IRQn_Type IRQn);

يتم ضبط NVIC Priority با ستخدام التابع HAL_NVIC_SetPriorityGrouping()، ثم يتم ا ستخدام التابع HAL_NVIC_SetPriority() لضبط مستوى الأولوية للمقاطعة المختارة.

لمعرفة إذا ماكان هناك مقاطعة معلقة نستخدم التابع التالي:

uint32_t HAL_NVIC_GetPendingIRQ(IRQn_Type IRQn);

لتفعيل Pending bit لمقاطعة معينة برمجياً نستخدم التابع التالي:

void HAL_NVIC_SetPendingIRQ(IRQn_Type IRQn);

لتصفير Pending bit برمجياً نستخدم التابع التالي:

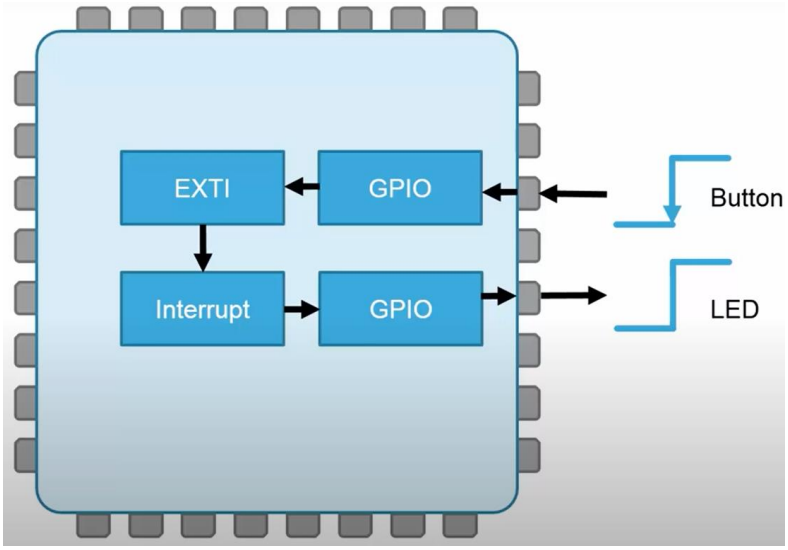
void HAL_NVIC_ClearPendingIRQ(IRQn_Type IRQn);

لمعرفة إذا كان هناك برنامج خدمة مقاطعة يتم تنفيذه حالياً نستخدم التابع التالي:

uint32_t HAL_NVIC_GetActive(IRQn_Type IRQn);

التطبيق العملي:

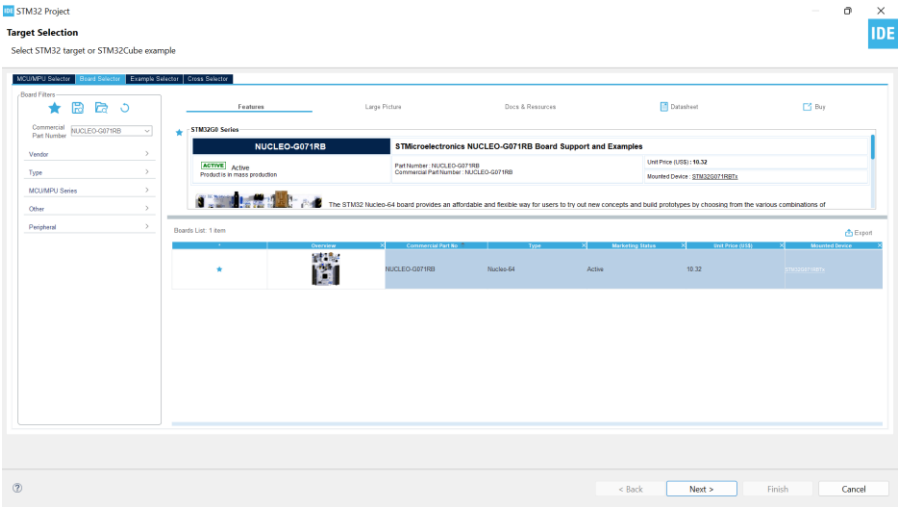
باستخدام لوحة Nucleo وبالاستفادة من المفتاح اللحظي الموصول مع القطب PC13 والليد الموصول على القطب PA5 ، قم بضبط إعدادات المتحكم STM32 وكتابة الكود المناسب كي يقوم الليد بعكس حالته المنطقية عند الضغط على المفتاح اللحظي PC13 مع مراعاة ضبط القطب PC13 كقطب مقاطعة خارجية.



الشكل (18):تطبيق عملي على مفهوم المقاطعة

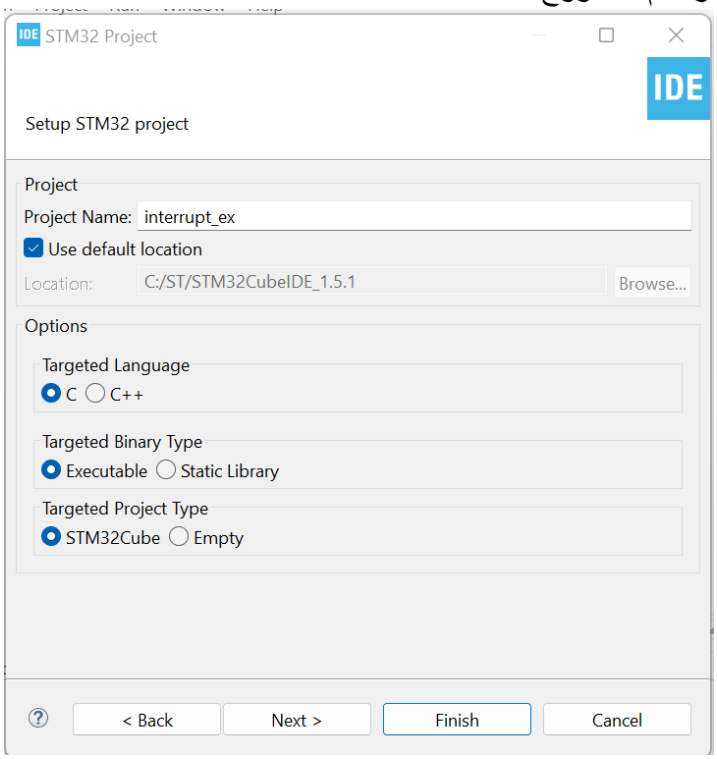
1. **ضبط إعدادات المشروع:**

الخطوة الأولى: فتح بيئة STM32CubeIDE وإنشاء مشروع جديد ثم اختيار لوحة Nucleo



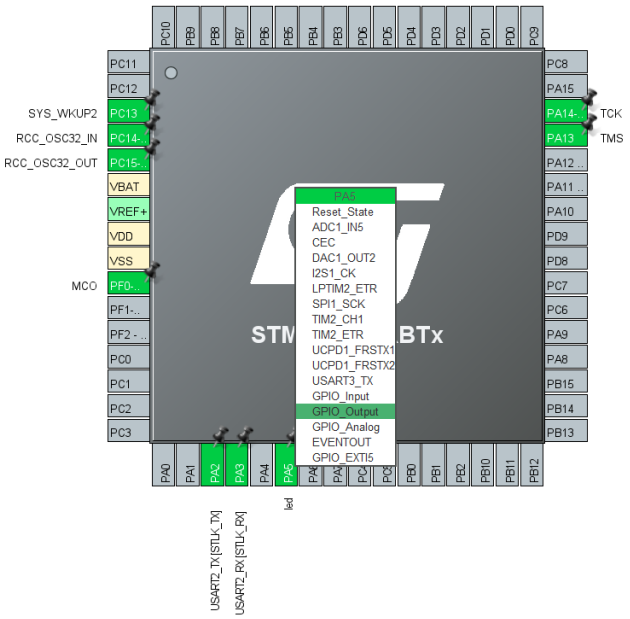
الشكل (19):فتح مشروع جديد واختيار اسم اللوحة

الخطوة الثانية: اختيار اسم للمشروع



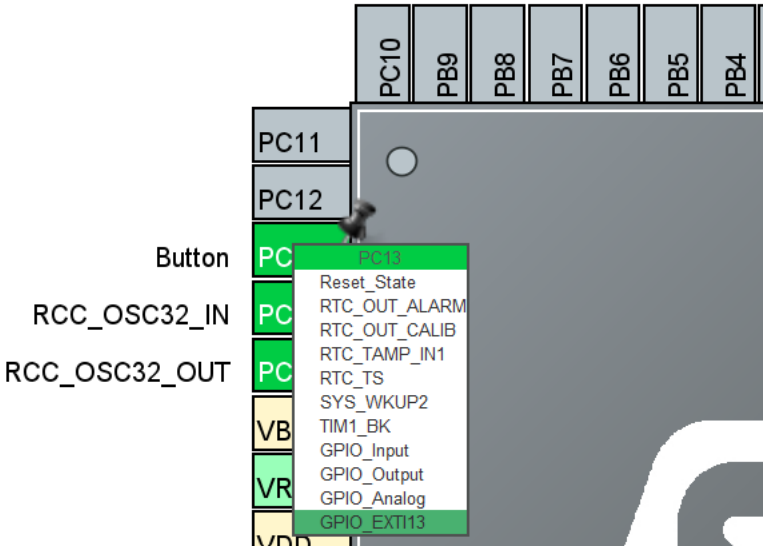
الشكل (20): اختيار اسم للمشروع

الخطوة الثالثة: اختر القطب PA5 لـ ضبطه كقطب خرج (هذا القطب مو صول معه داخلياً ليد ضمن لوحة Nucleo)



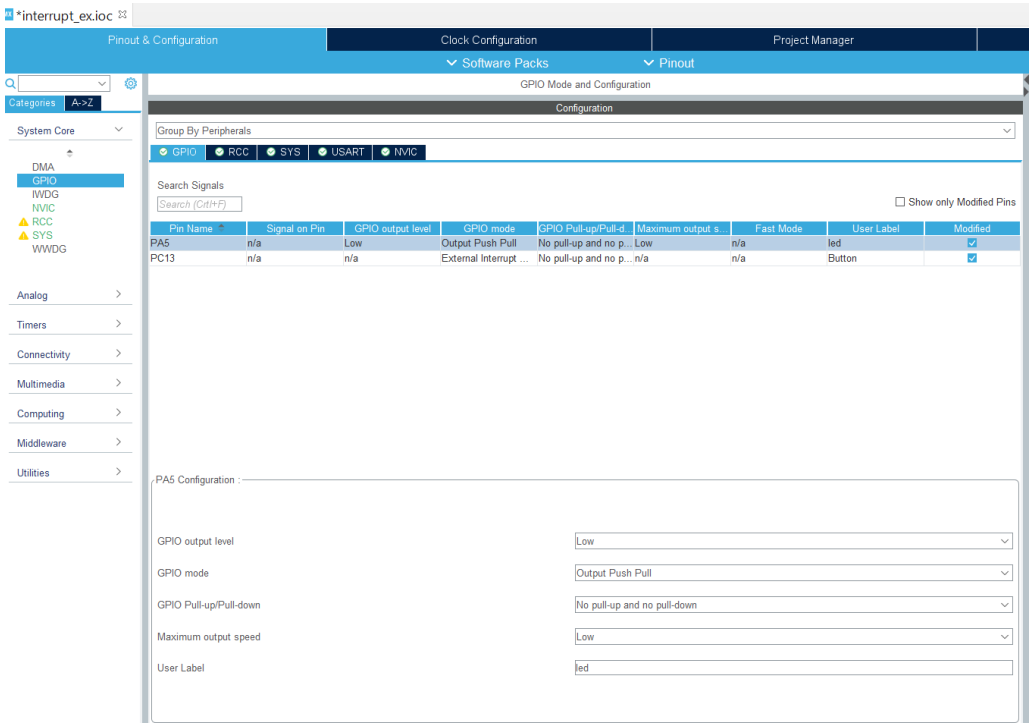
الشكل(21): ضبط القطب PA5 كقطب خرج

اختر القطب PC13 لـ ضبطه كقطب مقاطعة خارجية (هذا القطب مو صول معه داخلياً مفتاح لحظي ضمن لوحة Nucleo)



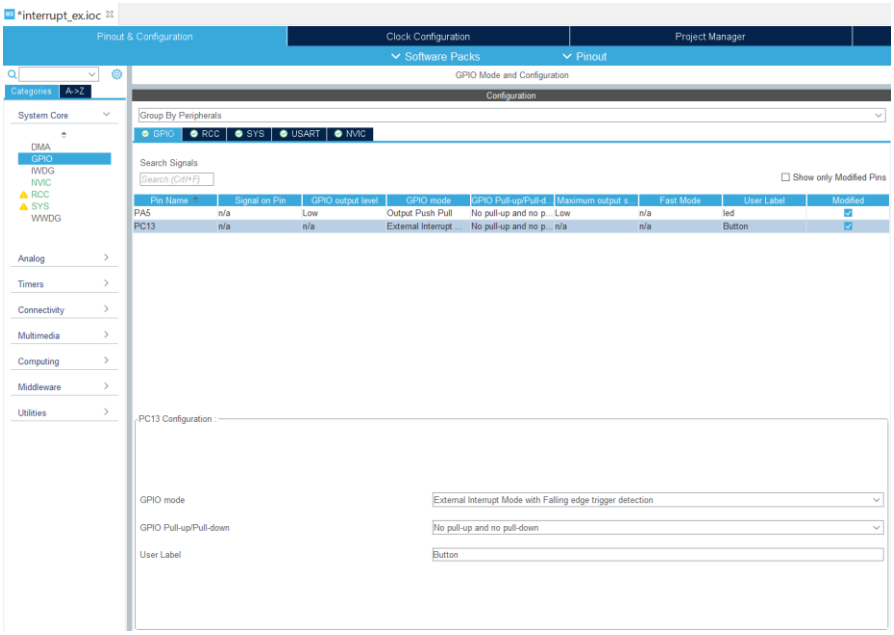
الشكل(21): ضبط القطب PC13 كقطب مقاطعة

الخطوة الرابعة: من Config Tab قم بضبط إعدادات قطب الخرج PA5 بالشكل التالي:



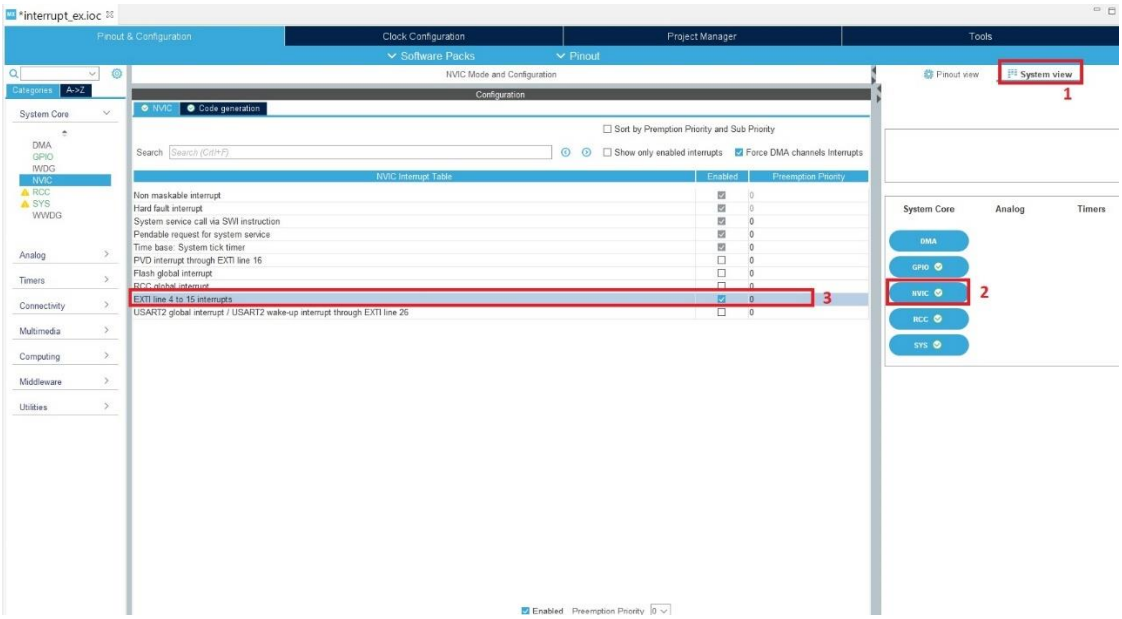
الشكل(22): ضبط إعدادات قطب الخرج PA5

وضبط قطب المقاطعة PC13 بالشكل التالي:



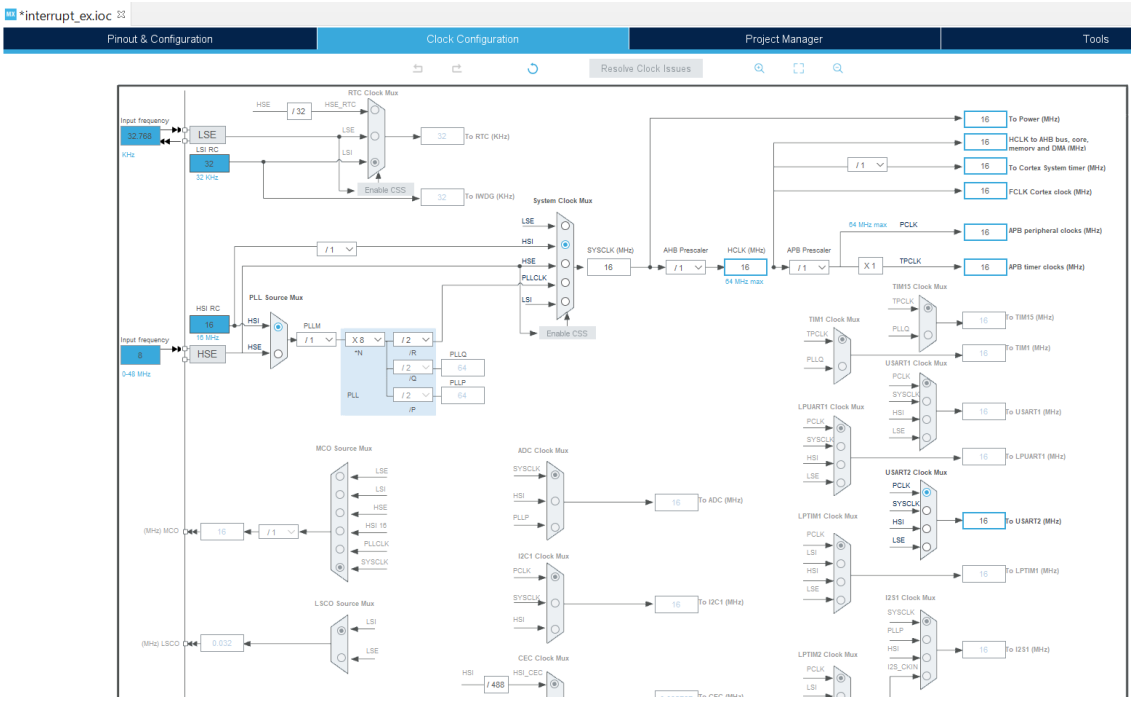
الشكل(23): ضبط إعدادات قطب المقاطعة PC13

الخطوة الخامسة: قم بفتح NVIC Tab ثم قم بتفعيل المقاطعات الخارجية، يمكنك أيضاً إعادة ضبط مستويات الأولوية للمقاطعات:



الشكل(24): تفعيل المقاطعات الخارجية

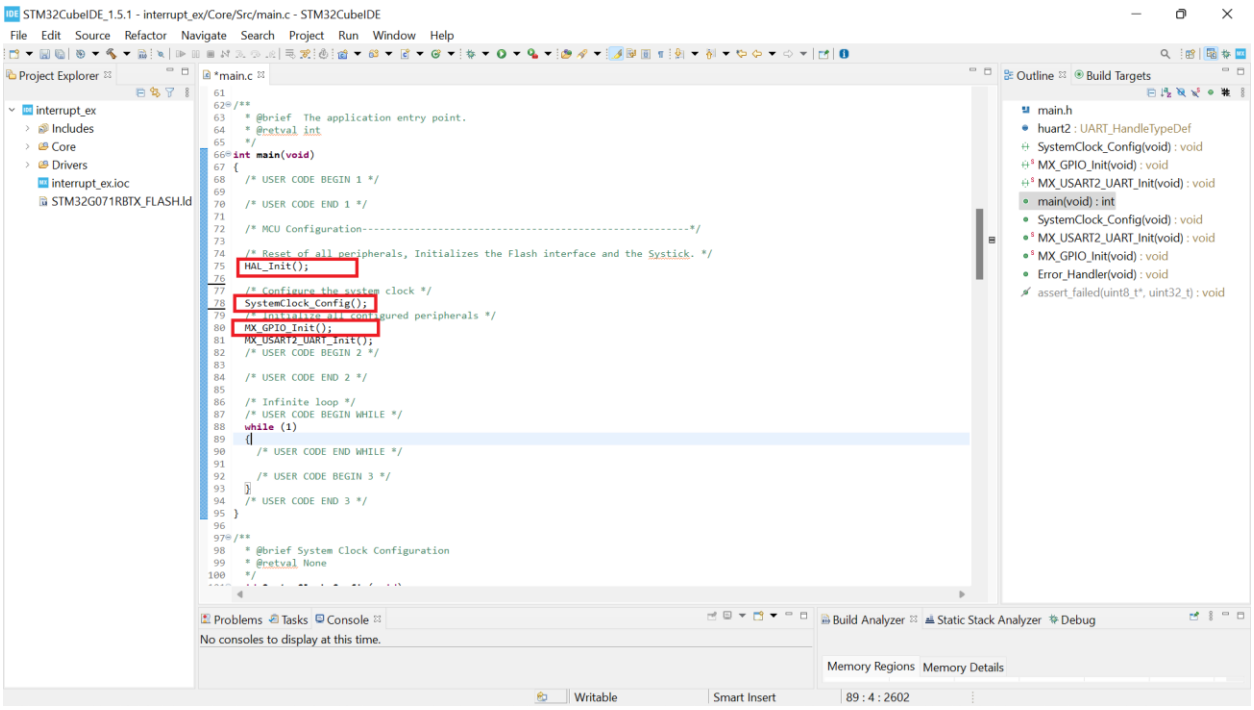
الخطوة السادسة: قم بضبط ساعة النظام على الساعة الداخلية(HSI) باعتبار أن لوحة Nucleo لا تحتوي على ساعة خارجية واختر التردد 64MHZ



الشكل(25): ضبط ساعة النظام على الساعة الداخلية(HSI)

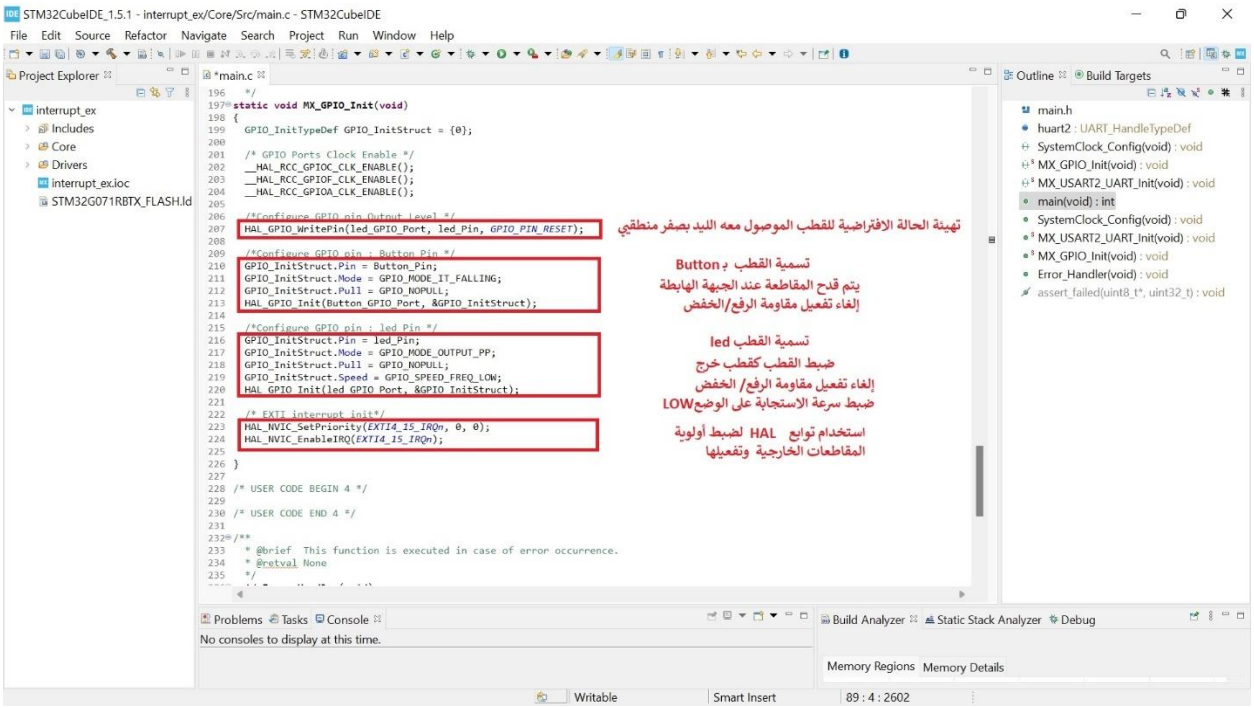
الخطوة الـ سابعة: قم بتوليد الكود اعتماداً على الإعدادات التي قمت بـ ضبطها من خلال الـ ضغط على زري Ctrl+S

حيث نلاحظ أن ملف الـ main.c - يحتوي على تهيئة وتضمين مكتبة HAL وضبط ساعة النظام بالإضافة إلى تهيئة المداخل والمخارج والطرفيات كما في الشكل التالي:



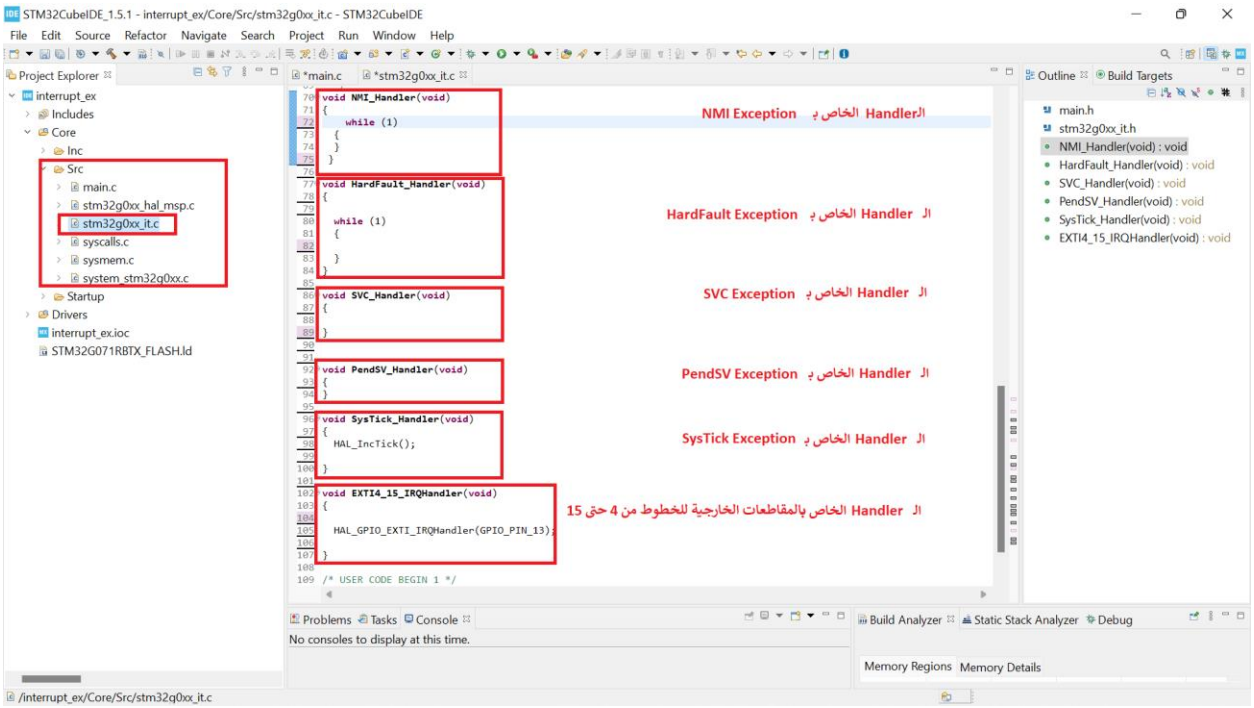
الشكل(26): الكود الناتج عن الإعدادات

حيث يتضمن التابع MX_GPIO_Init(void) التالي:



الشكل(27): التابع MX_GPIO_Init(void)

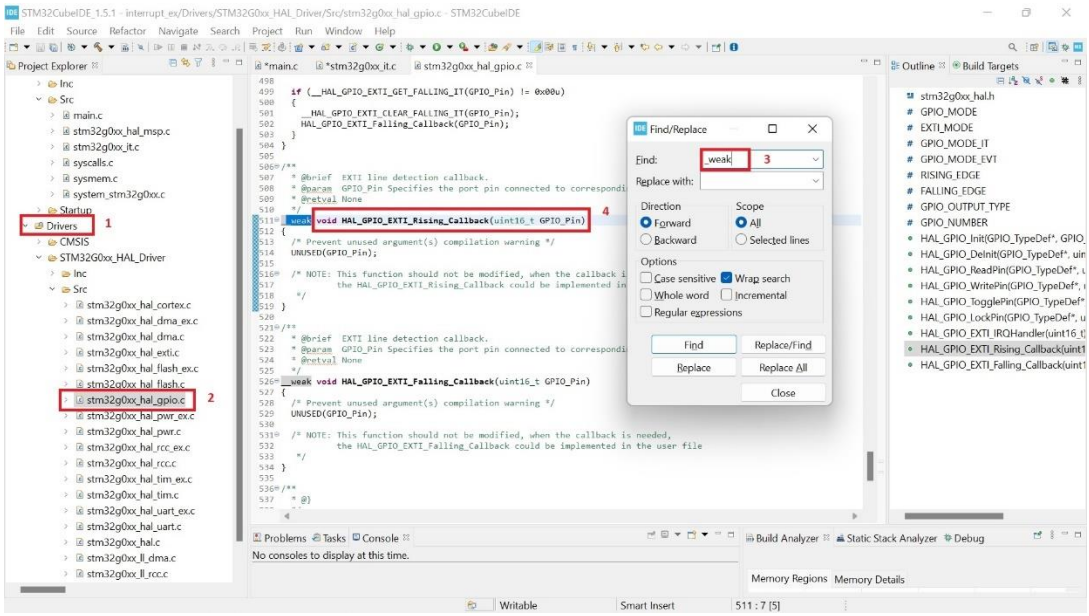
تقع الـ Handlers الخاصة بالاستثناءات/المقاطعات ضمن ملف stm32g0xx_it.c كما هو موضح بالشكل التالي



الشكل(28): الملف stm32g0xx_it.c

الخطوة الثامنة :

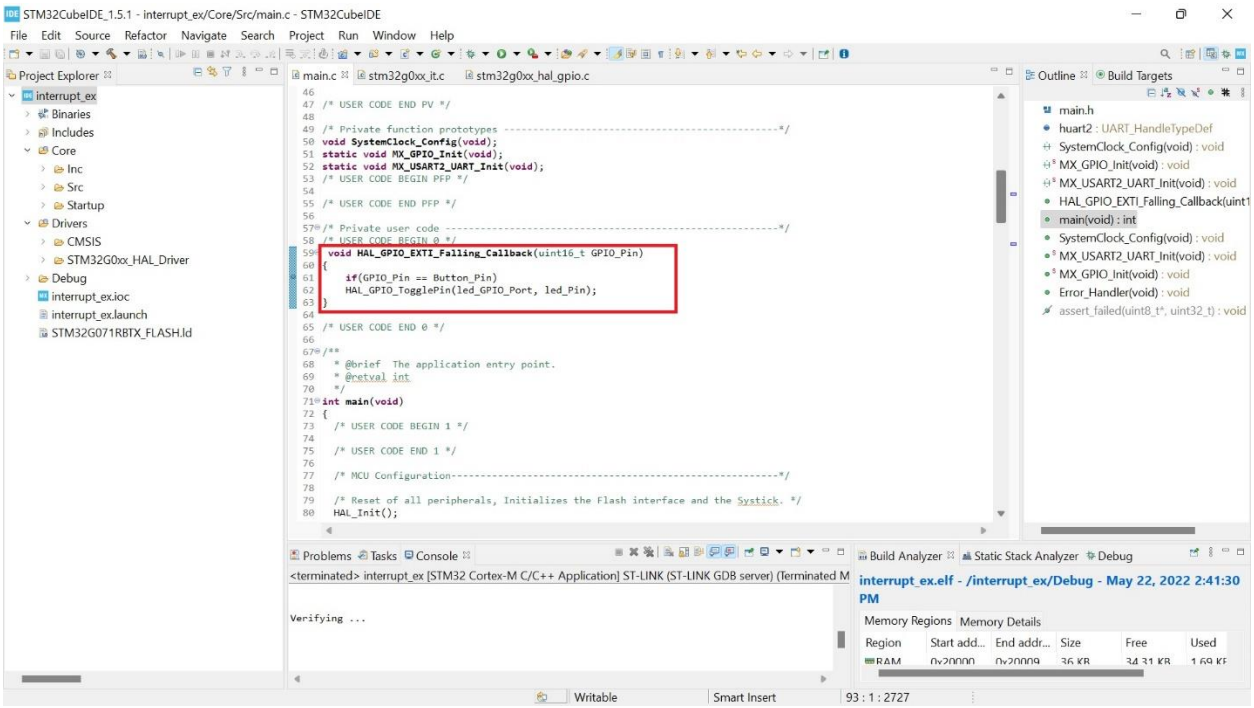
1- نختار المجلد Drivers
2- نقوم بفتح الملف stm32g0xx_hal_gpios.c
3- نبحث عن الـ function الذي يبدأ بكلمة weak_ ونحدد اسم الـ function وننسخه



الشكل(28):كيفية الوصول لبرنامج خدمة المقاطعة

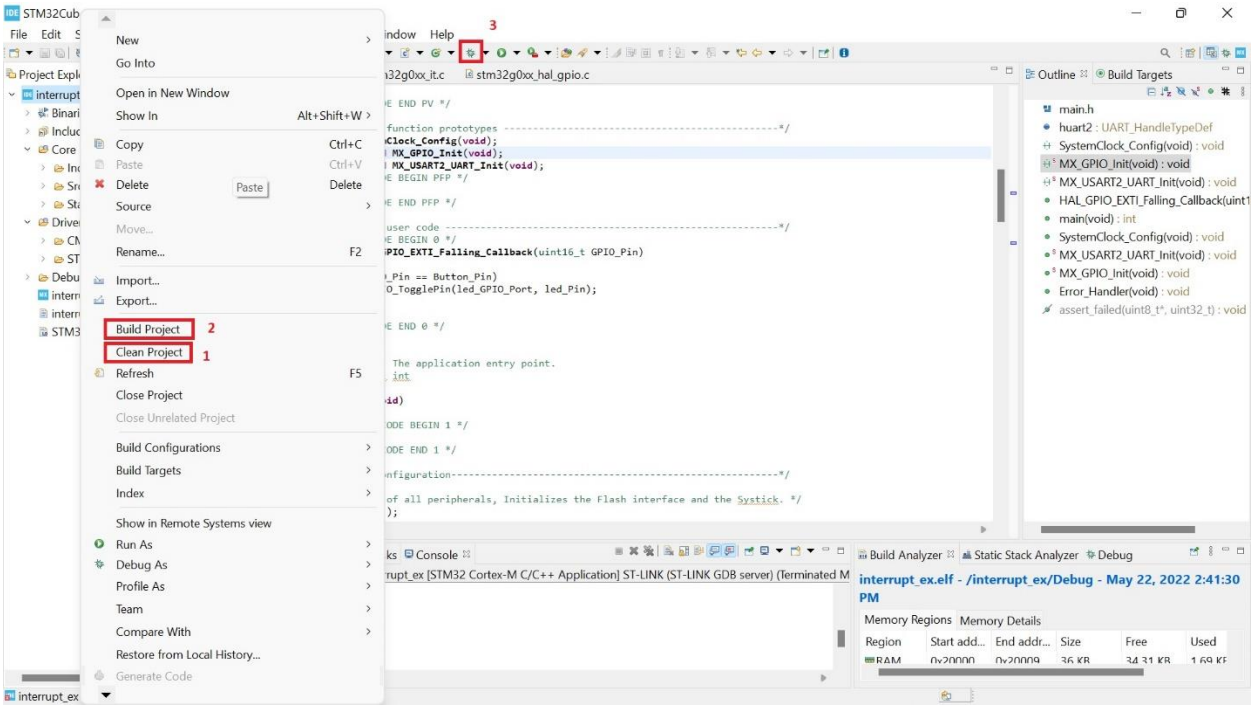
تعني كلمة weak_ أي سيتم استدعاؤه في حال لم يكتب المستخدم برنامج خدمة مقاطعة يحمل نفس الاسم.

4- نقوم بلصق الـ functionضمن ملف الـ main.c إما قبل أو بعد الـ int main() ونضع فيه الأوامر التي نريد تنفيذها عند طلب المقاطعة مثلاً هنا قمنا بفحص القطب الذي حدثت عنده المقاطعة ثم قمنا باستدعاء تابع HAL الذي يقوم بعكس الحالة المنطقية لليد عند حدوث المقاطعة.



الشكل(29):برنامج خدمة المقاطعة

الخطوة التاسعة: التأكد من خلو الكود من أي أخطاء برمجية من خلال تنظيف الكود clean project ثم ترجمة الكود من خلال Build Project ثم رفع الكود للمتحكم (متحكم لوحة Nucleo) ومراقبته عبر Debug.



الشكل(30):ترجمة الكود ورفعه وفتح جلسة debug

الكود بالكامل:

```
#include "main.h"

 void HAL_GPIO_EXTI_Falling_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == Button_Pin)
        HAL_GPIO_TogglePin(led_GPIO_Port, led_Pin);
}

int main(void)
{
   HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
    while (1)
  {

  }

}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

   HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);
   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSIDiv = RCC_HSI_DIV1;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the peripherals clocks
  */
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(led_GPIO_Port, led_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Button_Pin */
  GPIO_InitStruct.Pin = Button_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(Button_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : led_Pin */
  GPIO_InitStruct.Pin = led_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(led_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI4_15_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);

}
void Error_Handler(void)
{
    __disable_irq();
  while (1)
  {
  }
  }

#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{
  }
#endif
```

ملاحظة:

لزيادة سرعة الاستجابة للمقاطعات عليك الاستغناء عن مكتبة HAL .

أيضاً لايمكن استخدام التأخير الزمني من مكتبة HAL ضمن برنامج خدمة المقاطعة.