Interruptions

Sommaire

- Définition du problème
- Qu'est ce qu'une interruption ?
- Comment gérer les interruptions dans le STM32 ?
- Comment programmer les interruptions ?
- Cas pratiques

Définition du problème

- Les évènements touchant un microcontrôleur sont imprévisibles
  - Il sont dit : asynchrones
  - Pas de connaissance *a priori* sur leur instant d'apparition

- Le déroulement d'un programme est séquentiel et synchrone
  - Besoin de méthode pour capturer ces évènements externes ou internes
  - Comment faire ?

Définition du problème

- Solution : la boucle de scrutation
  - Attente active (polling)

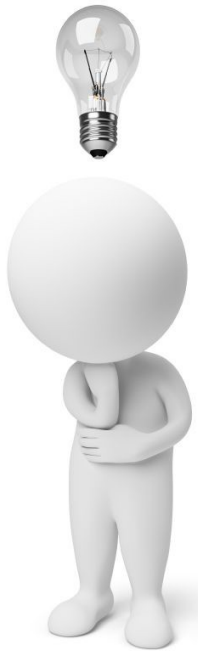
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    while(1)
    {
        avance();
        if (voiture == stop)
        while (!plus_voiture)
            if (voitures == true)
                plus_voiture=0;
        }
    return 0;
}
```

Définition du problème

- Problèmes : boucle de scrutation
  - On ne peut pas régler le temps d'attente de manière précise.
  - Le microprocesseur ne peut rien faire d'autre !
    - 100% du temps processeur est alloué à la tâche de scrutation
    - Le temps alloué à l'évènement est potentiellement long
    - Pas de multi-tâches possible
      - Pas de possibilité de faire d'autres opérations pendant l'attente

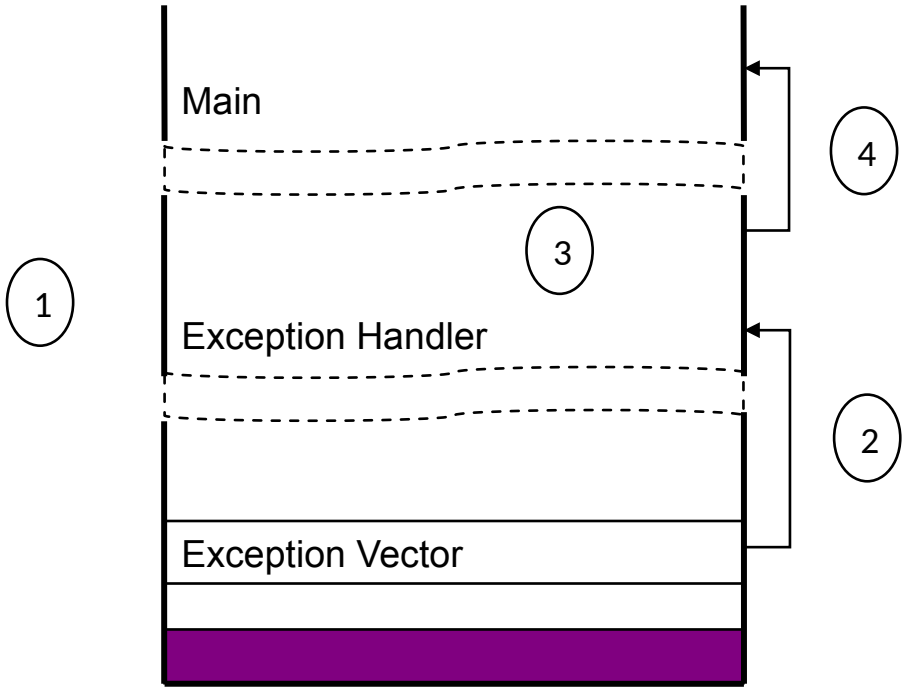
Définition du problème



- Pour mettre en œuvre efficacement le multitâche, il faut que le processeur ne soit pas bloqué en attente des opérations d'entrée-sorties, qui doivent se dérouler en parallèle avec les calculs.
- La gestion des périphériques doit être confiée à des circuits spécialisés. Des commandes (requêtes) seront envoyées au processeur pour le prévenir par un signal d'interruption.
- Le fonctionnement par interruption décharge ainsi le processeur de la surveillance des périphériques. Le processeur peut du coup consacrer son temps à l'avancement des autres tâches.

Interruption : c'est quoi ?

- Une interruption est un événement interne ou externe qui va de manière chronologique :
  1. Arrêter le fonctionnement en cours du programme.
  2. Sauvegarder le contexte d'exécution : état du programme en cours d'instruction (registres et compteur ordinal – program counter)
  3. Exécuter une suite d'instruction pour servir l'interruption : routine de service d'interruption – programme spécifique à l'interruption
  4. Restaurer le contexte précédent d'exécution
  5. Reprendre le fonctionnement du programme



Interruption : classification

- Une interruption est un événement interne ou externe : 2 types

- **Interruptions internes : exceptions**
  - Les interruptions internes sont appelées exceptions. Elles surviennent lors d'une erreur d'exécution du programme (overflow, watchdog, instruction réservée, adressage, etc.)

- **Interruptions externes : interruptions**
  - Les interruptions externes concernent essentiellement les demandes de service des périphériques. Elles arrivent de manière asynchrone avec les instructions en cours d'exécution.

- **Propres à l'exécution d'un processus**
- **Concernent des appels systèmes**
- **Erreurs d'exécution**

- **Commandée par les périphériques**
- **Asynchrone / aux instructions en cours**

Architecture du Cortex M4

- Le STM32F446 possède plus de 100 lignes d'interruptions et exceptions systèmes
  - 16 lignes d'exceptions systèmes
    - Reset
    - NMI (Non-masquable Interrupts)
    - SysTick (Timer)
    - Fault, …
  - 91 interruptions liées aux périphériques
    - GPIO : broches d'entrée / sortie
    - Bus : SPI, I2C
    - Périphériques d'acquisition et restitution : ADC et DAC

Interruption : hiérarchisation

Table des exceptions et interruptions

exceptions

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault	All class of fault	0x0000_000C
	0	settable	MemManage	Memory management	0x0000_0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048

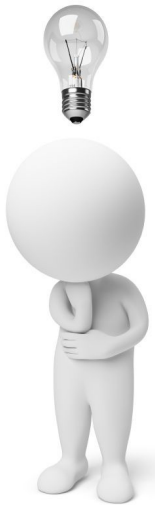
exceptions

Interruptions

Position	Priority	Type of priority	Acronym	Description	Address
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	USB_HP_CAN_TX	USB High Priority or CAN TX interrupts	0x0000_008C
20	27	settable	USB_LP_CAN_RX0	USB Low Priority or CAN RX0 interrupts	0x0000_0090
21	28	settable	CAN_RX1	CAN RX1 interrupt	0x0000_0094
22	29	settable	CAN_SCE	CAN SCE interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4

Interruption : hiérarchisation

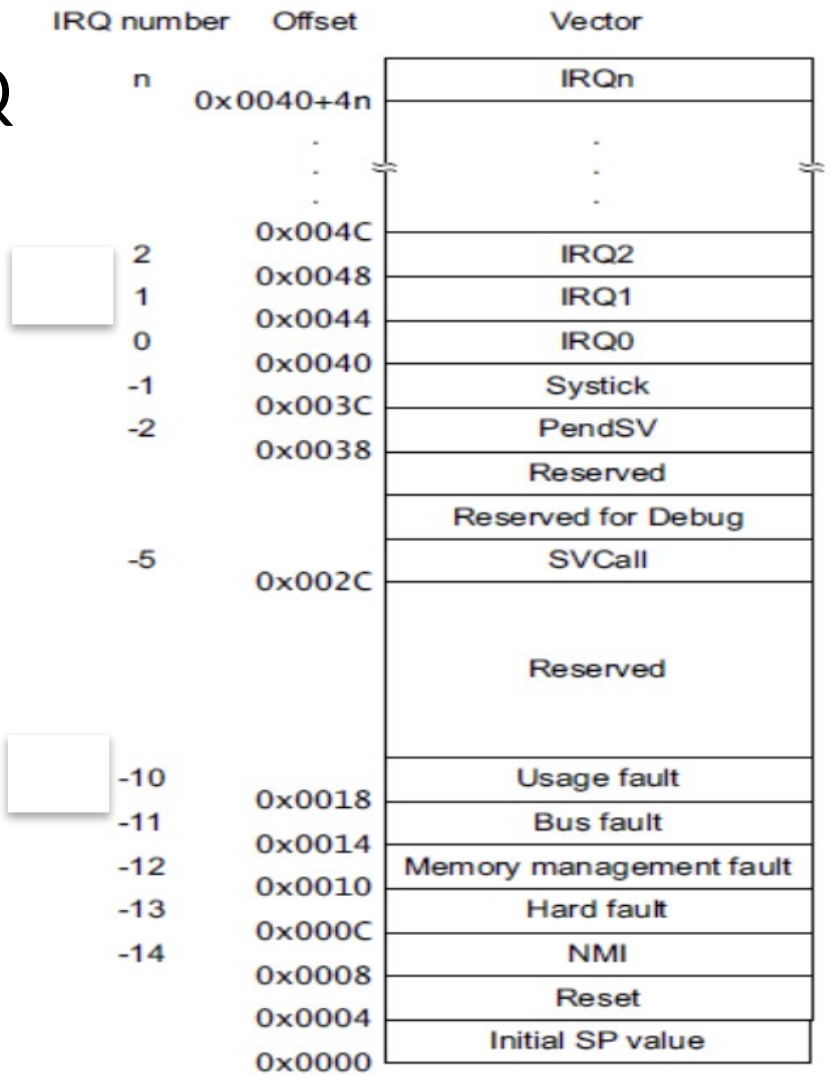
- Si deux interruptions arrivent en même temps, comment fait on ?



- Hiérarchisation
  - Niveaux de priorité
    - IRQ : Interruption Request
  - Table des vecteurs d'interruption
  - Contrôleur d'interruption
    - Organisation des services

Interruption : hiérarchisation

- Exemple de hiérarchisation et d'IRQ
  - Fixée par un plan mémoire
  - IRQ1 plus prioritaire que l'IRQn

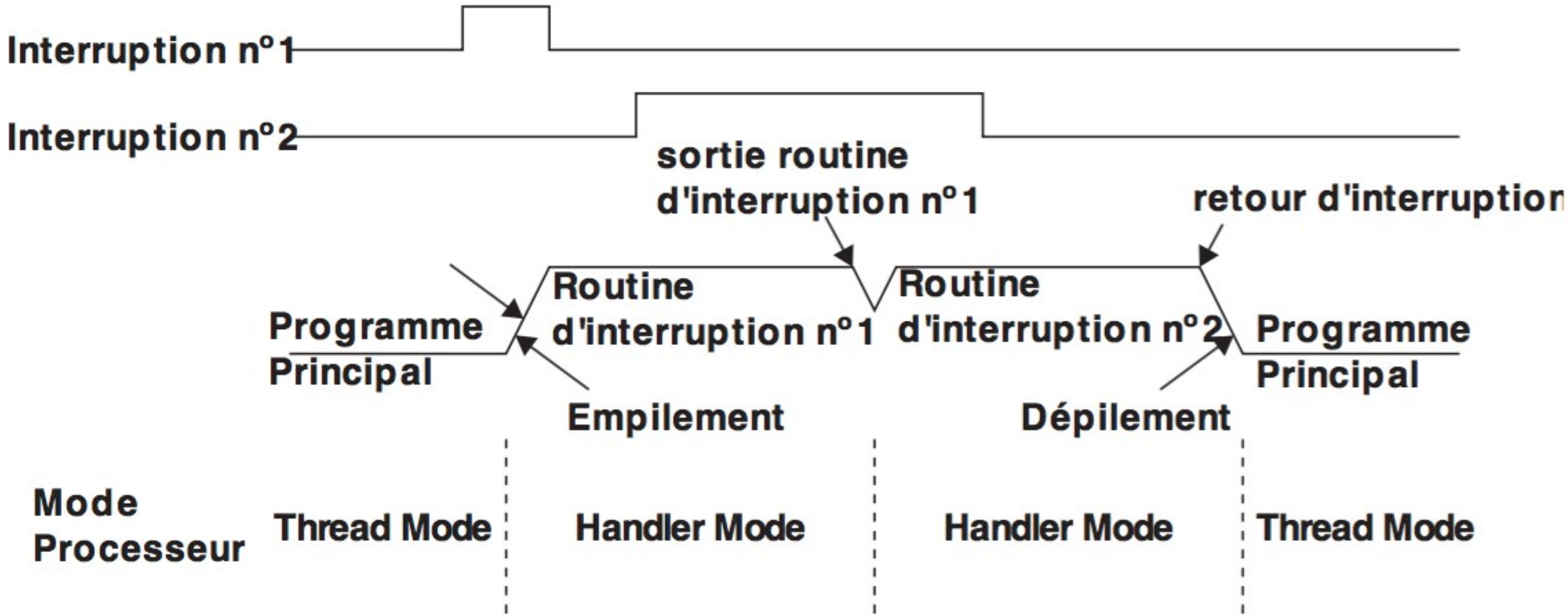


Interruption : hiérarchisation

- La hiérarchisation repose sur une politique de gestion des interruptions
- Elle doit permettre qu'une interruption plus prioritaire puisse interrompre une interruption moins prioritaire en cours d'exécution.
- Elle doit interdire qu'une interruption moins prioritaire ne puisse pas interrompre une interruption plus prioritaire en cours d'exécution.
- Elle peut fixer des priorités non modifiables
- Elle permet de fixer les niveaux de priorités de la plupart des sources d'interruption
  - Souplesse / à l'utilisateur

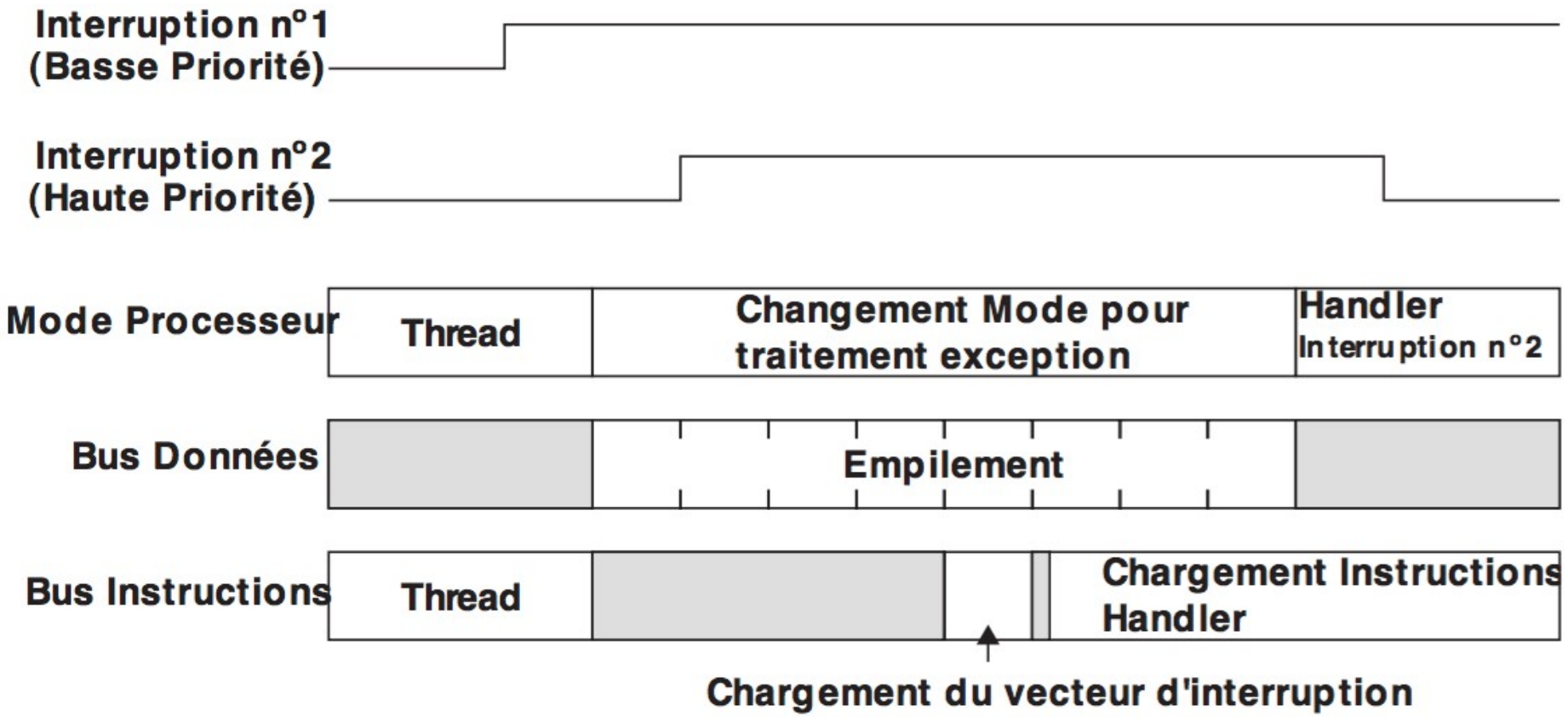
Interruption : typologie

- Niveau Interruption 1 > Niveau Interruption 2



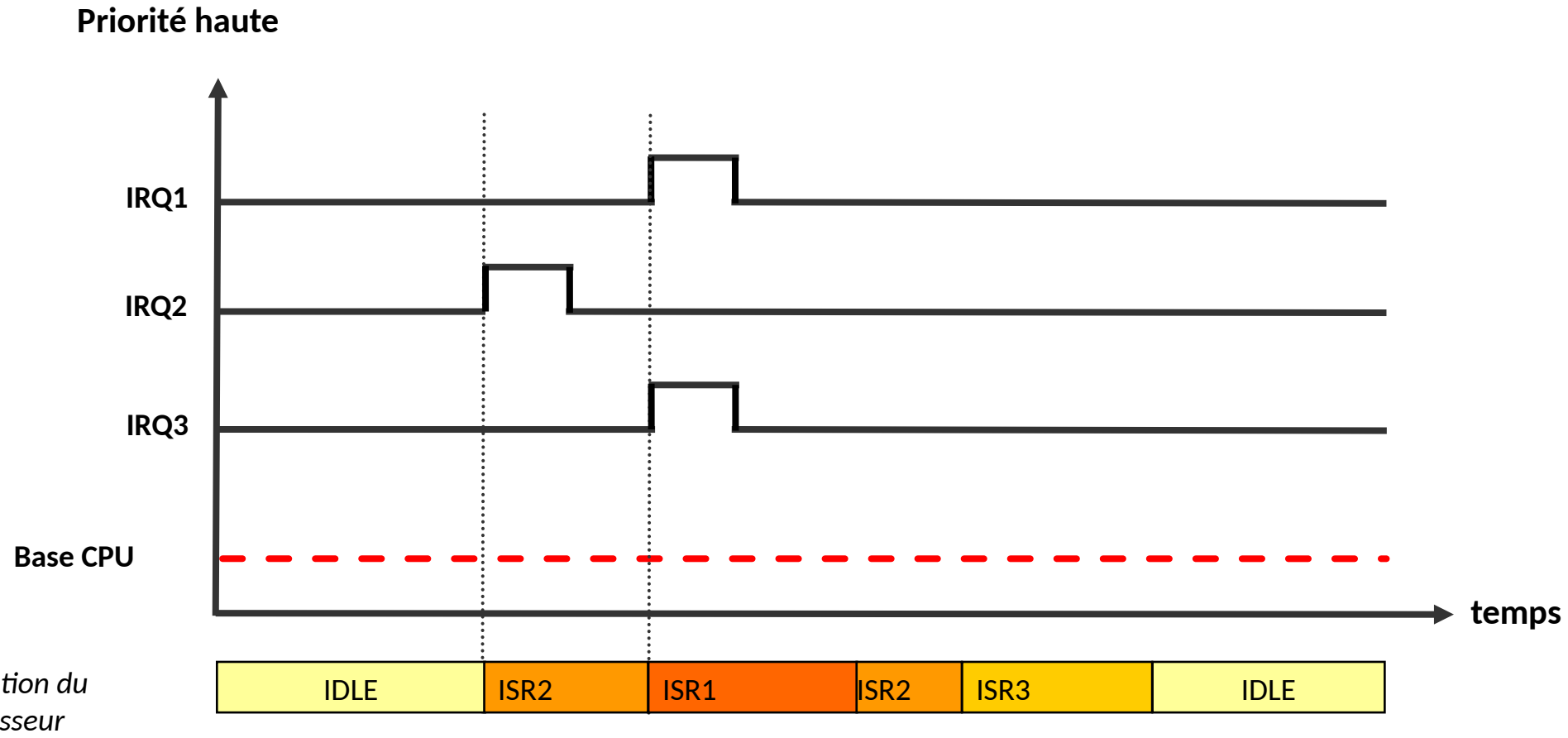
Interruption : typologie

- Niveau Interruption 1 > Niveau Interruption 2



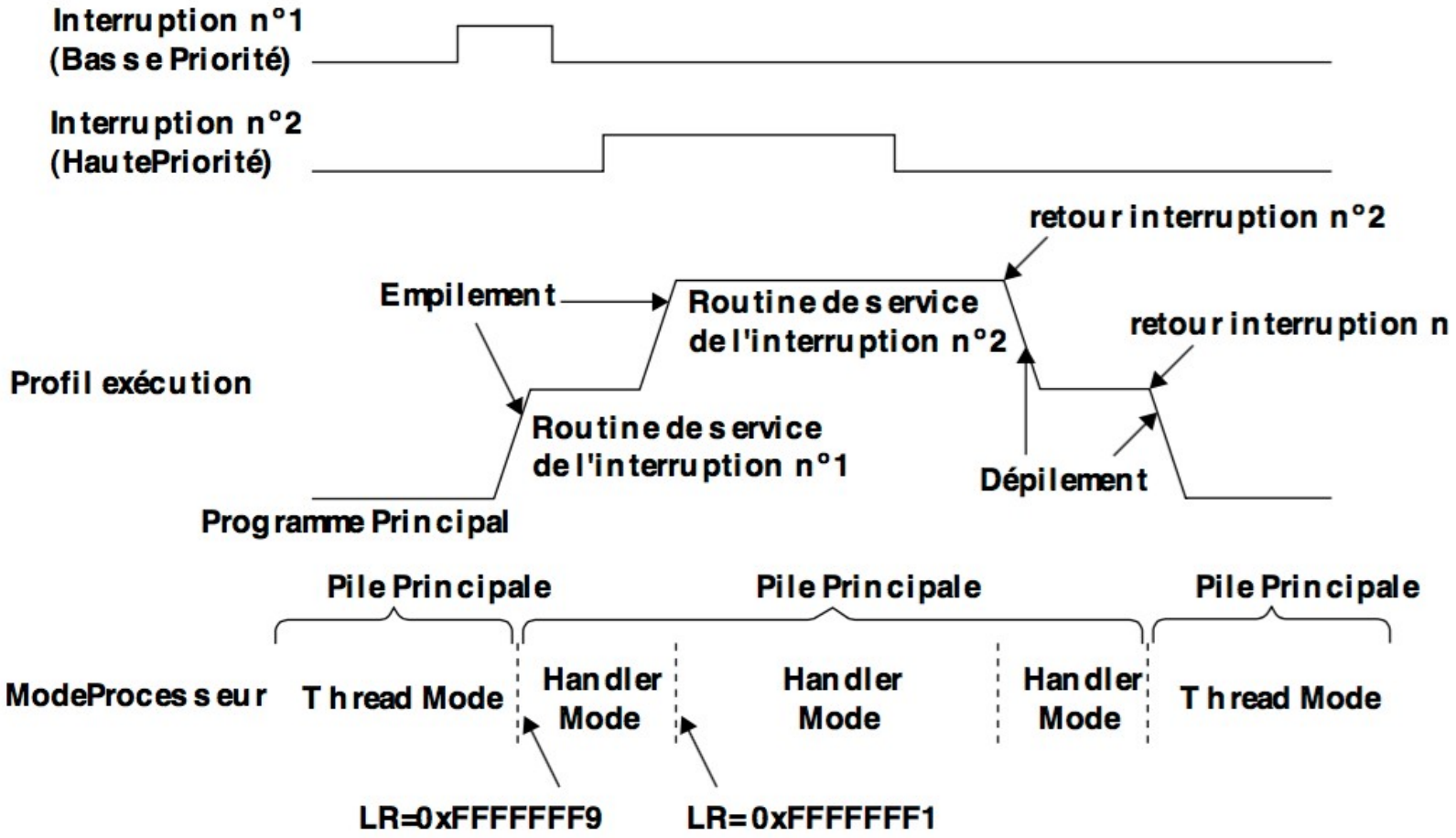
Interruption : typologie

- Cas de trois interruptions



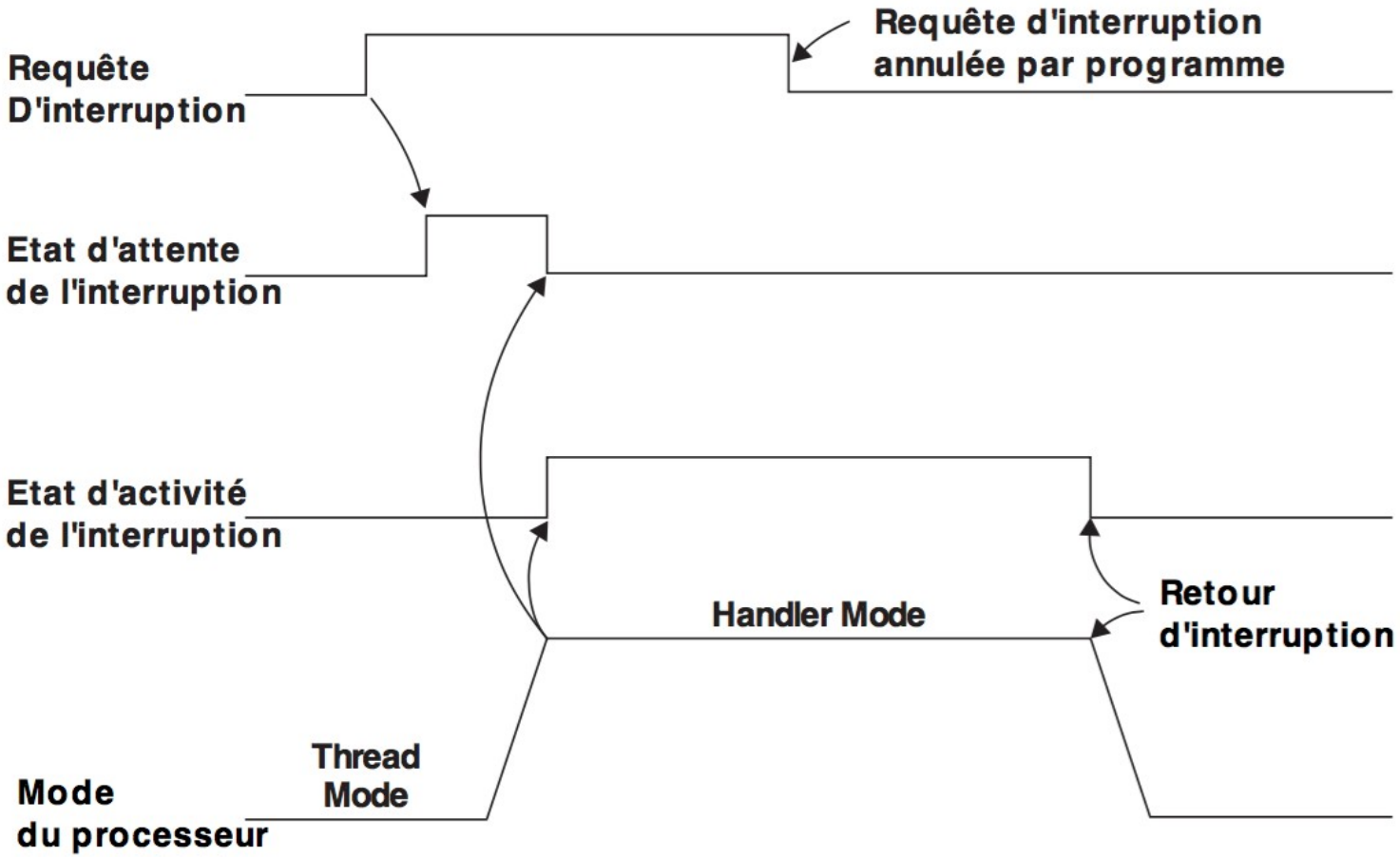
Interruption : typologie

- Niveau Interruption 2 > Niveau Interruption 1



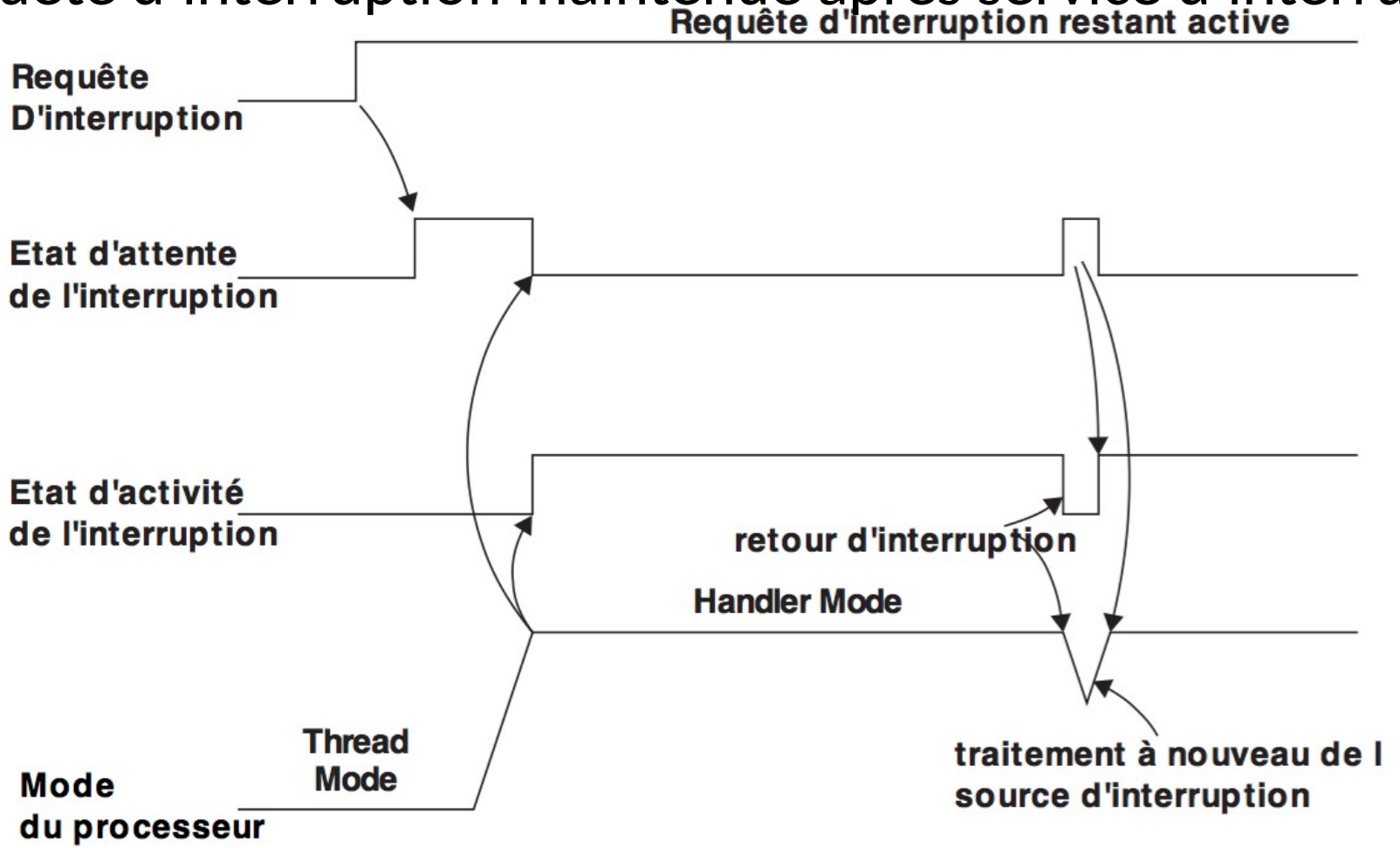
Interruption : typologie

- Requête d'interruption maintenue lors du début d'interruption



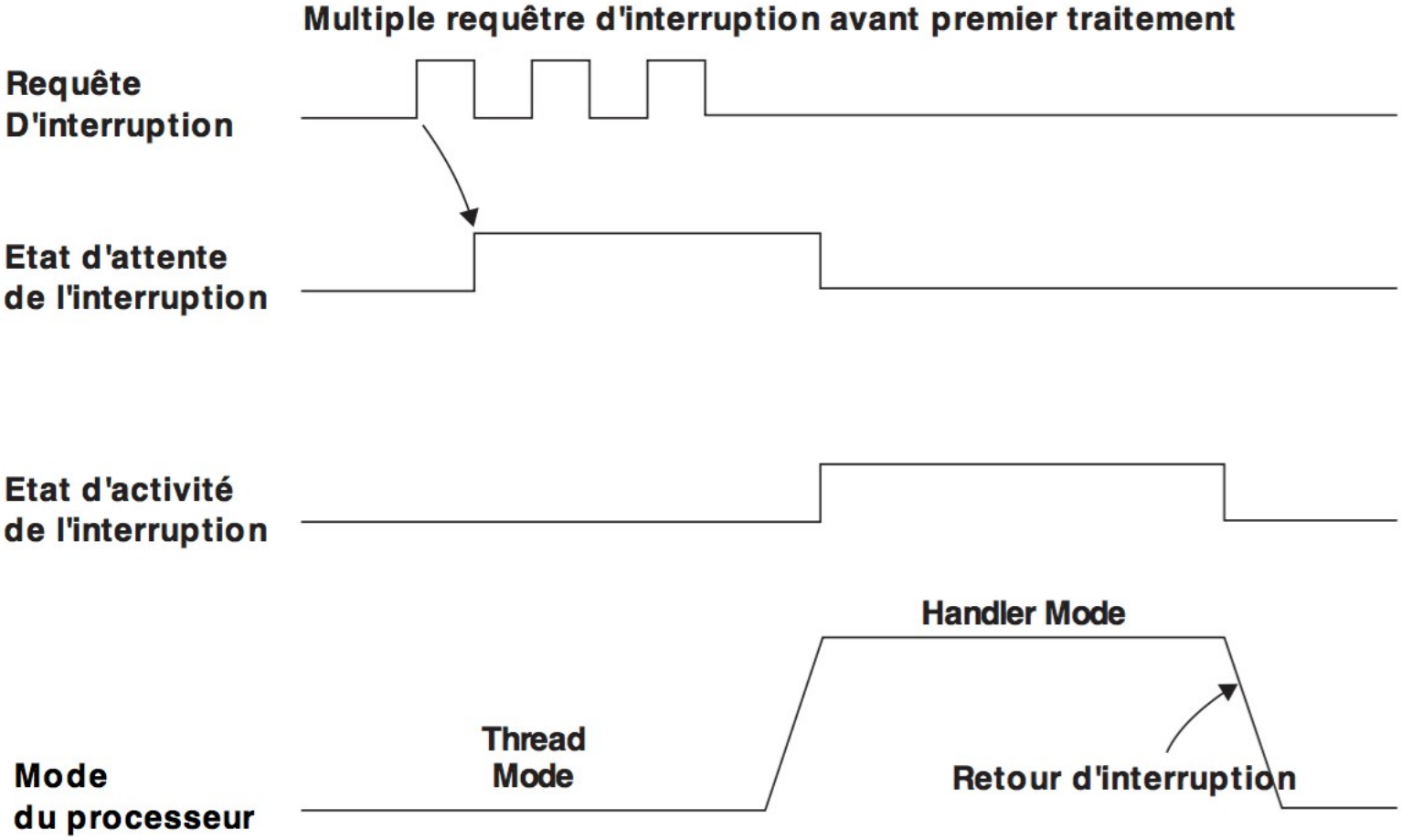
Interruption : typologie

- Requête d'interruption maintenue après service d'interruption



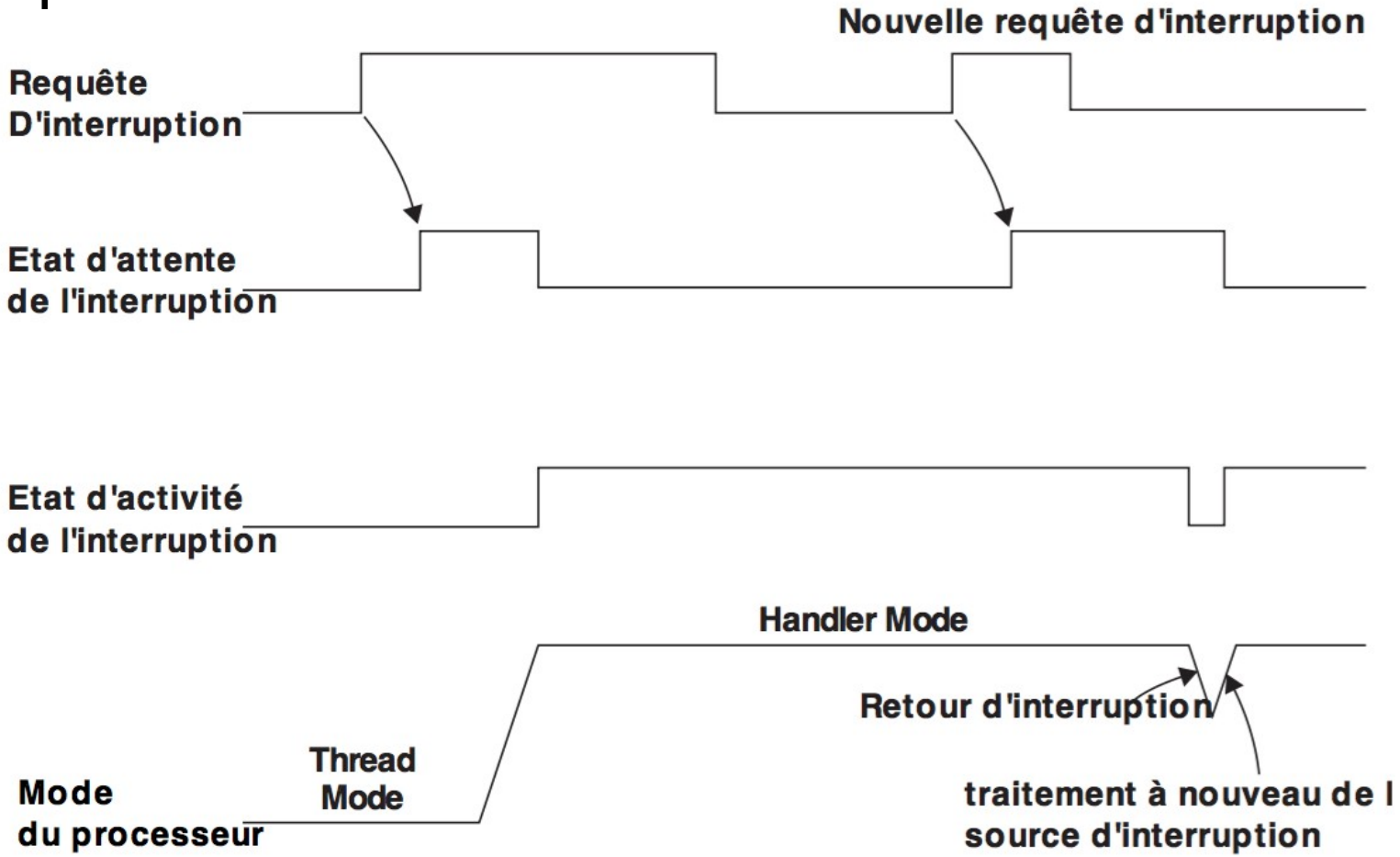
Interruption : typologie

- Requêtes multiples



Interruption : typologie

- Requête pendant le service



Interruptions : masquage ou non traitement

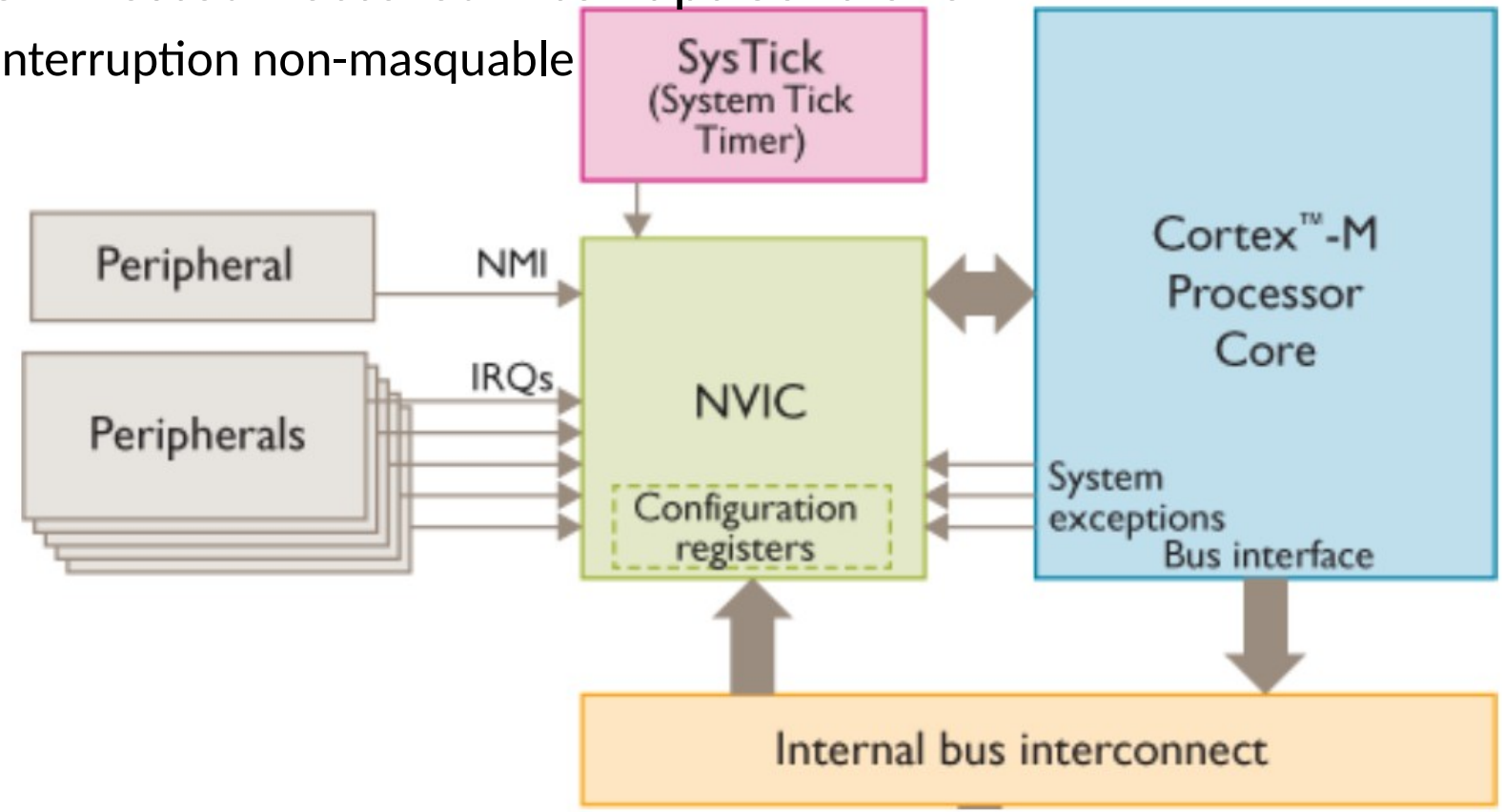
- Masquage des interruptions
  - Retardement de la prise en compte
    - Nécessite une hiérarchisation des interruptions de priorité inférieure
  - Mémorisation des interruptions masquées
    - Notion de Pile
      - FIFO pour les interruptions

- Désarmer les interruptions
  - Essentiellement les interruptions non systèmes

Gestion des interruptions dans le STM32F

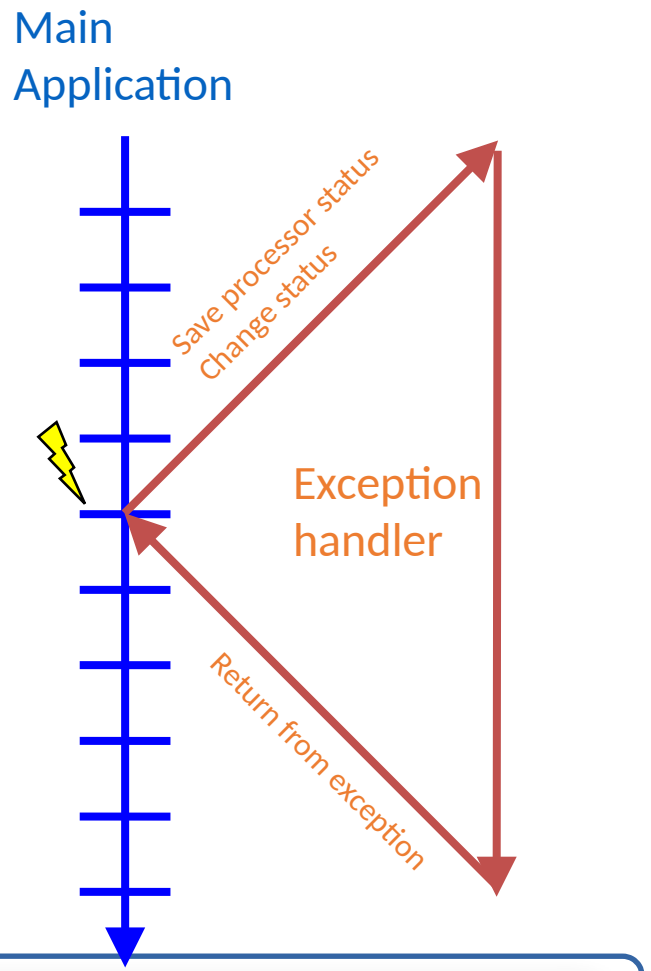
Architecture du Cortex M4

- Gestion des interruptions au niveau architecture
  - Contrôleur NVIC – Nested Vectored Interrupt Controller
    - Supporte une interruption non-masquable



Architecture du Cortex M4

- Les interruptions du STM32 sont dites « vectorisées »
  - Quand une interruption intervient, la « bonne » routine doit être fournie : **Interrupt Handler**
  - Pour cela il faut **déterminer l'adresse du programme** correspondant à cette routine
  - Cette information est stockée dans une **table des vecteurs d'interruption**
  - Par défaut cette table est stockée à l'adresse 0
  - Cette table est relogeable via le registre VTOR (Vector Table Offset Register), c'est à dire qui peut être mis en mémoire à un autre endroit)
  - La table est organisée en mots de 4 octets (32 bits)
  - L'adresse du vecteur d'une interruption est calculée en multipliant par 4 son numéro.



Architecture du Cortex M4

- Table des vecteurs d'interruption

Adresse	Numéro de l'exception	Valeur
0x00000000	–	Valeur initiale de MSP
0x00000004	1	Valeur initiale du Compteur de Programme (vecteur de Reset)
0x00000008	2	Adresse de la fonction de gestion de l'exception NMI
0x0000000C	3	Adresse de la fonction de gestion de l'exception HardFault
...	...	Autres adresse des fonctions de gestion des exceptions

Architecture du Cortex M4

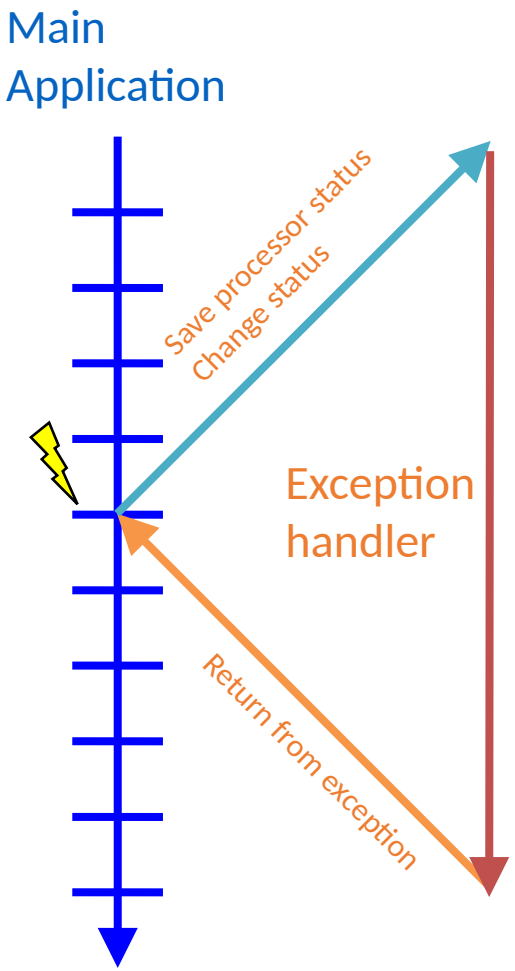
- Registre Vector Table Offset – VTO

Bits	Nom	Type	Valeur initiale	Description
29	TBLBASE	R/W	0	Base de la table : 0 pour ROM, 1 pour RAM
28:7	TBLOFF	R/W	0	Valeur du décalage de la table des vecteurs d'exceptions

Architecture du Cortex M4

- Traitement d'une interruption par le Cortex M4 comporte trois phases :
  - Phase 1 : Sauvegarde du contexte du programme
  - Phase 2 : Chargement de la routine d'interruption – Interrupt Handler
  - Phase 3 : Mise à jour du pointeur de pile SP (Stack Pointer), du registre de lien LR (Link Register) et du compteur ordinal PC (compteur de programme)

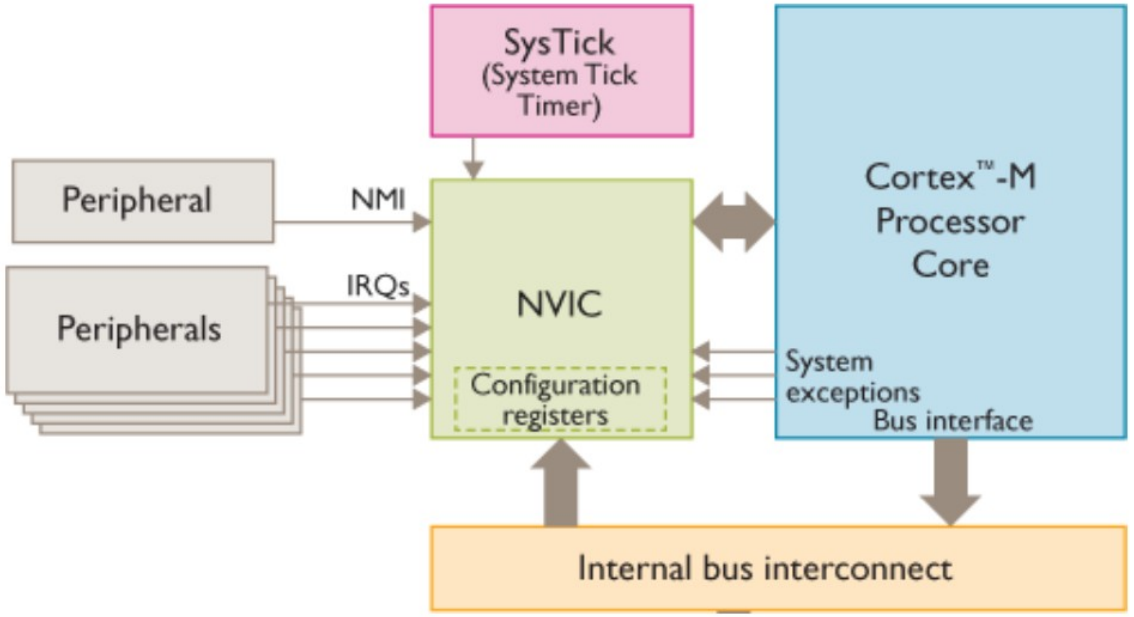
Architecture du Cortex M4



- **Phase 1 : Sauvegarde du contexte du programme**
  - Registrement des registres xPSR, PC, LR, R12, R3-R0
  - Sauvegarde de CPSR dans SPSR
  - Sauvegarde de PC dans LR
- **Phase 2 : Chargement de la routine d'interruption – Interrupt Handler**
  - Changement du mode du processeur
  - PC = adresse du service d'interruption
  - Exécution du code utilisateur
- **Phase 3 : Mise à jour** du pointeur de pile SP (Stack Pointeur), du registre de lien LR (Link Register) et du compteur ordinal PC (compteur de programme)
  - Dépilement des registres après le service de l'interruption
  - Restauration du contexte du programme avant l'interruption (xPSR, PC, LR, R12, R3-R0)
  - Restauration de SPSR dans CPSR
  - Restauration de PC à partir de LR

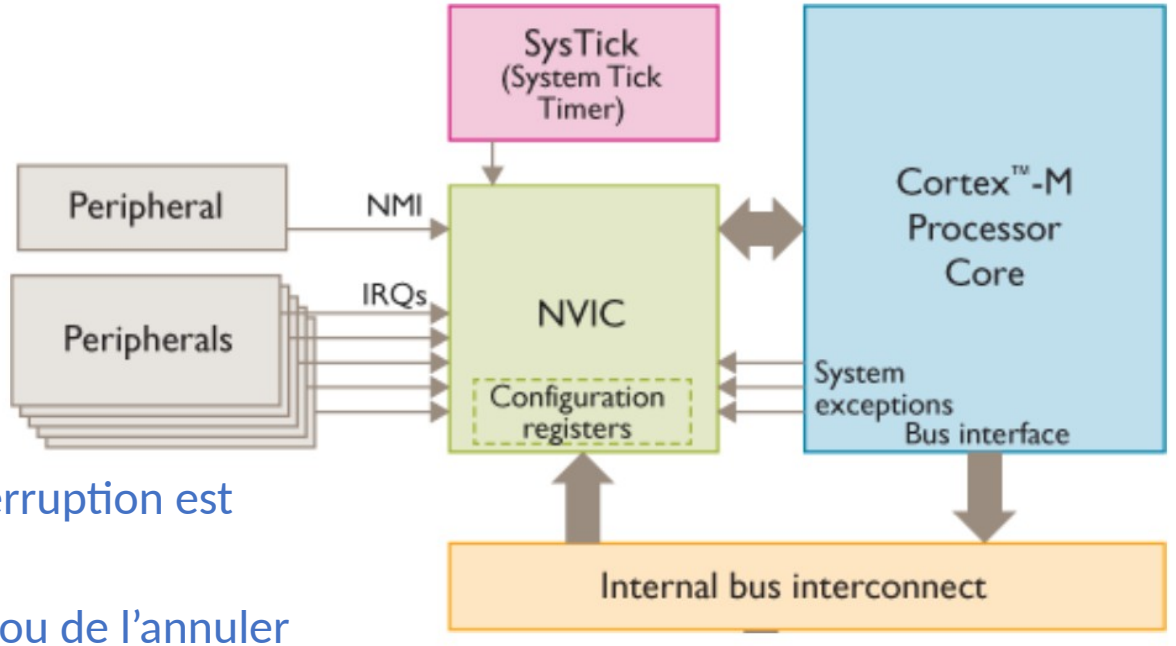
Contrôleur NVIC

- Contrôleur NVIC – Nested Vectored Interrupt Controller
- Le NVIC permet de contrôler 82 sources d'interruptions provenant des périphériques
- Le NVIC interrompt le CPU avec l'IRQ de plus haute priorité
  - Le CPU utilise le numéro d'IRQ pour accéder à la routine d'interruption via son adresse en mémoire (table d'interruption)



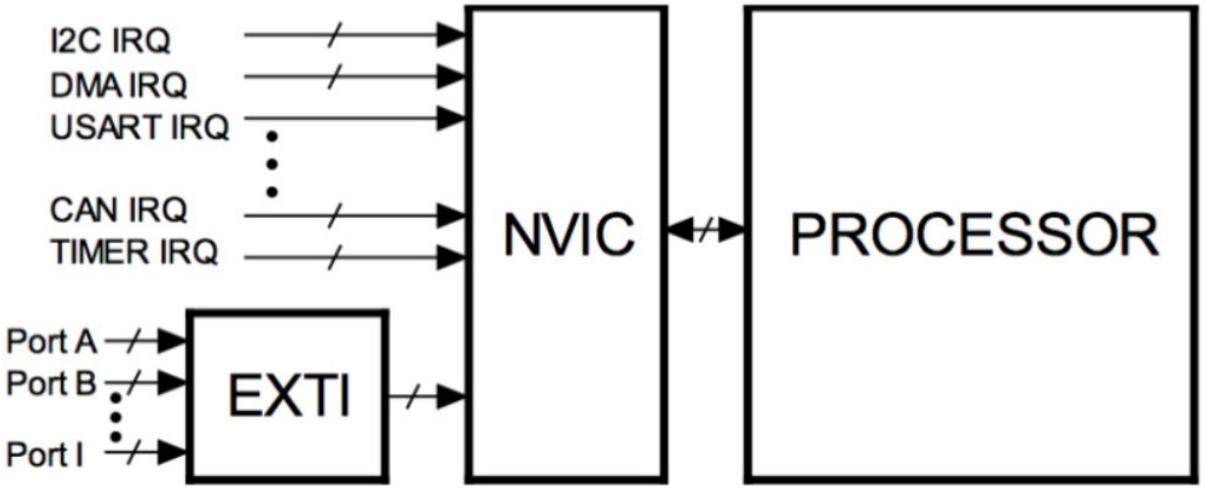
Contrôleur NVIC

- Les registres du NVIC
- NVIC_ISERx/NVIC_ICERx
  - Interrupt Set/Clear Enable Register
  - 1 = Enable interruption / Clear
  - Chaque IRQ a son propre bit d'activation
- NVIC_ISPRx/NVIC_ICPRx
  - Interrupt Set/Clear Pending Register
  - Lecture d'un 1 dans le registre ISPR permet de savoir si l'interruption est suspendue (pending state)
  - En mode écriture ⥤ permet de suspendre une interruption ou de l'annuler
- NVIC_IABRx – Interrupt Active Bit Register
  - Permet de savoir si l'interruption est active



Interruptions externes

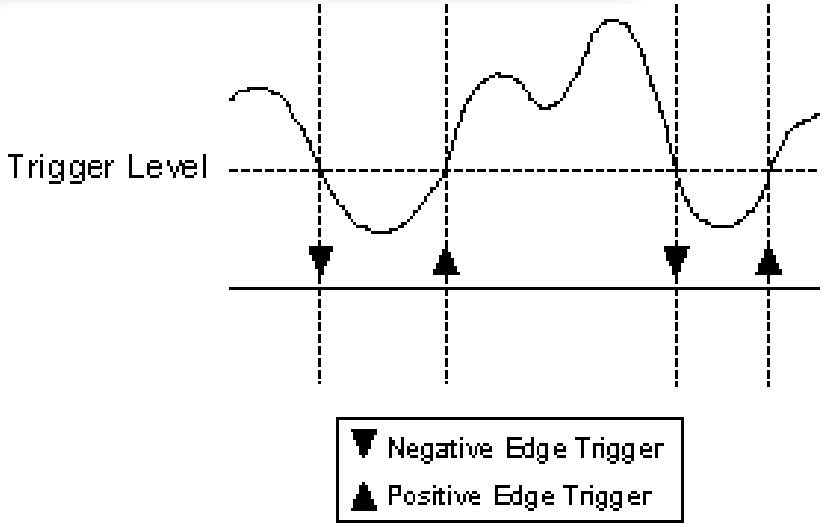
- Gestion des interruptions externe au niveau architecture
  - EXTI proviennent généralement de lignes externes – connectées à un GPIO
  - 23 détecteurs de front pour synchroniser les évènements et les interruptions délivrés par 240 broches de GPIO et 7 évènements internes

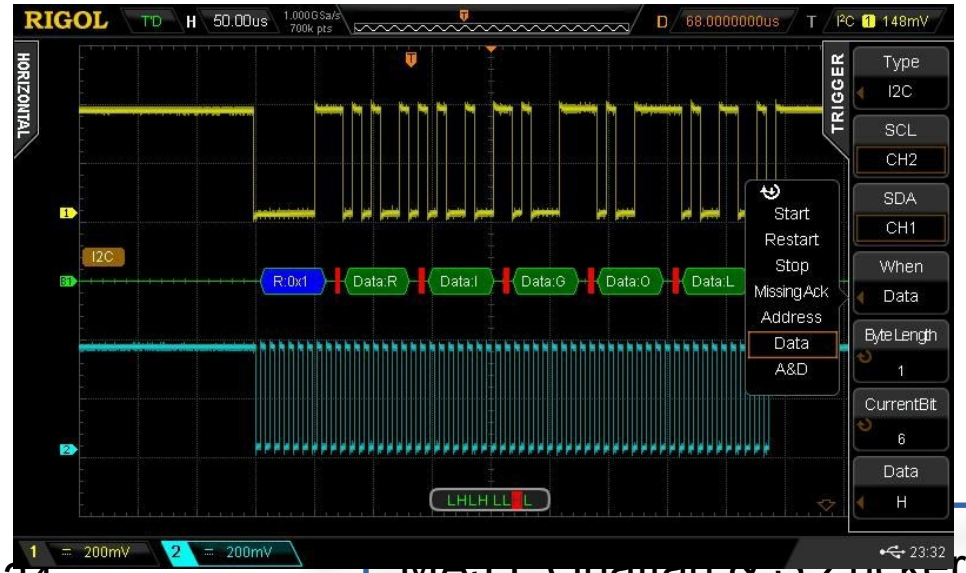


Exemples d'application des interruptions

Interruptions sur GPIO

- Interruption sur GPIO
- Exemple : signal de trigger
  - Synchronisation d'une action sur un signal extérieur
  - Synchronisation sur un front ou un niveau
  - Appui sur un bouton, clavier, etc.
  - Analyse d'un signal, d'un bus, etc.





STM Cube

Exemple de configuration des GPIOs

- Démonstration de la configuration des GPIOs sous STMCubeMx

1. Configuration des périphériques
2. Génération du code
3. Importation sous keil
4. Execution du code

Interruptions sur GPIO

- Interruption sur GPIO avec l'environnement mbed

- Exemple : déclenchement d'une action à partir de l'appui d'un bouton

```
#include "mbed.h"

InterruptIn mybutton(USER_BUTTON);
DigitalOut myled(LED1);

float delay = 1.0; // 1 sec

void pressed()
{
    if (delay == 1.0)
        delay = 0.2; // 200 ms
    else
        delay = 1.0; // 1 sec
}

int main()
{
    mybutton.fall(&pressed);
    while (1) {
        myled = !myled;
        wait(delay);
    }
}
```