# Microcontrôleur ARM Cortex M3

## TD 3 : Gestion des interruptions

*Ce TD doit vous permettre de comprendre comment sont gérées les interruptions du microcontrôleur STM32 et en particulier :*

- *Les interruptions sur mise à jour (update) d'un Timer,*
- *Application aux déplacement d'une balle sur écran LCD.*

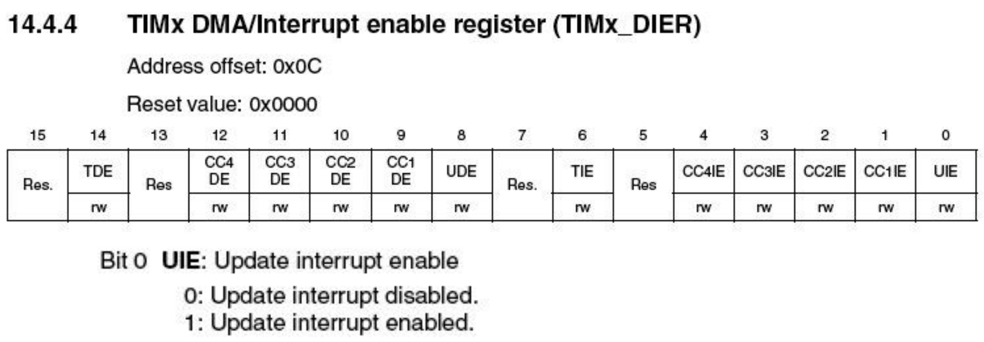
**Exercice 1 :** Interruptions du Timer 1 (TIM1)

*A partir de l'exemple de programme ci-dessous :*

```
/*  Includes  -----------------------------------------------------------------
*/
#include "stm32f10x.h"        // STM32F10x Library Definitions

#define UIE ………………..//à compléter
#define UIF (1<<0)
#define CEN (1<<0)

#define LEDS_PORTB   GPIOB->ODR

#define SETENA0 *(volatile unsigned long *)0xE000E100
#define TIM1_UP_IRQChannel (1<<25)

void Enable_GPIO(void)
{
      RCC->APB2ENR |= (1 << 3); // Enable GPIOB clock

}

void Init_GPIO(void)
{
      GPIOB->CRH = 0x33333333; // Mode=0b11 (50MHz) et CNF=0b00 (Push-Pull)

}
```

```
/* Configuration Timer 1 -----------------------------------------------------
------------*/
void cfgTimer1(void)
{
    RCC->APB2ENR |= (1 << 11);
    TIM1->PSC = 450;
    TIM1->ARR = 64000;
    TIM1->DIER |= UIE;
    TIM1->CR1 |= 0x0001;

    SETENA0 |= TIM1_UP_IRQChannel;

}

/* Traitement Interruption ---------------------------------------------------
--------------*/
void TIM1_UP_TIM10_IRQHandler (void)
{
    if(TIM1->SR & UIF)
    {
    LEDS_PORTB ^= (1<<8);
      TIM1->SR &= ~UIF;
    }
}

/* Programme principal -------------------------------------------------------
----------*/
int main(void)
{
  Enable_GPIO();
  Init_GPIO();
  cfgTimer1();

  while(1)
  { }
}
```

- *Analyser la configuration du Timer1 et déterminer la période à laquelle le drapeau UIF sera levé*
- *Préciser le rôle du bit UIE (registre TIM1->DIER : voir documentation ci-dessous)*
- *Dans le programme, compléter la ligne définissant l'équivalence de UIE*



**14.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)**

Address offset: 0x0C

Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | TDE | Res | CC4 DE | CC3 DE | CC2 DE | CC1 DE | UDE | Res. | TIE | Res | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
|  | rw |  | rw | rw | rw | rw | rw |  | rw |  | rw | rw | rw | rw | rw |

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.

1: Update interrupt enabled.

*La table des vecteurs d'interruptions donne le numéro et l'adresse affectés à une source d'interruption (cf document de cours)*

- *A partir de cette table des vecteurs, repérer l'interruption déclenchée par le mécanisme de mise à jour (auto-reload) du Timer1 ?*
- *Quelle est sa position et son adresse dans la table des vecteurs d'interruptions ?*
- *Quelle information devra-t-on trouver à l'adresse en question ? De combien d'octets a-t-on besoin pour stocker cette information ?*

*Le fichier de démarrage (startup)* ***startup_stm32f10x_xl.s*** *du microcontrôleur que nous utilisons en TP permet, entre autre, d'initialiser cette table des vecteurs (voir annexe). Il va permettre de faire le lien direct entre la position de l'interruption dans cette table et le nom (donc l'adresse) de la routine d'interruption.*

- *Quel devra être le nom de la routine d'interruption relative à la fonction d'auto-reload du Timer1 ?*

*Au niveau du NVIC (coeur Cortex), il faut préciser quels seront les canaux d'interruptions externes autorisés à déclencher une interruption. Cette validation se fait par les registres suivants :*

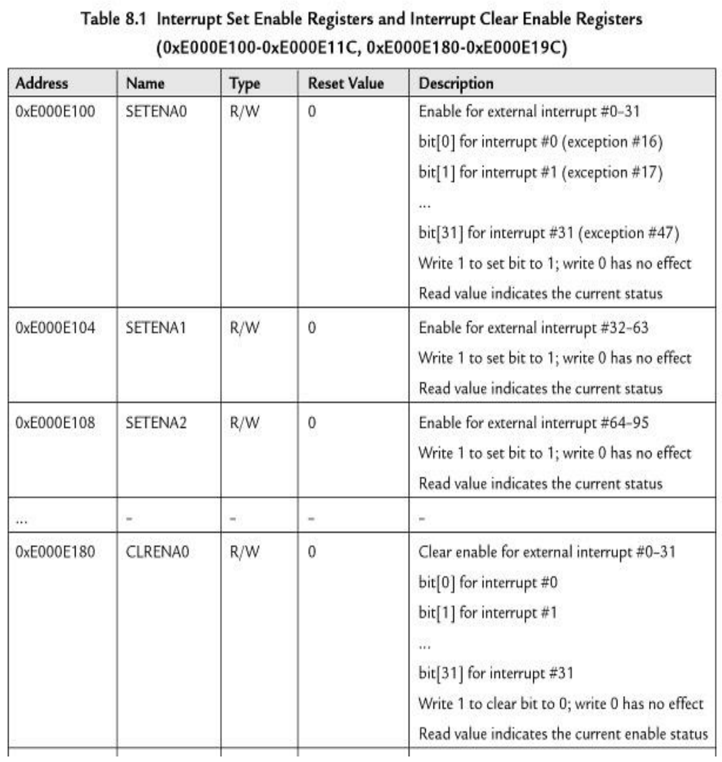


Table 8.1 Interrupt Set Enable Registers and Interrupt Clear Enable Registers (0xE000E100-0xE000E11C, 0xE000E180-0xE000E19C)

| Address | Name | Type | Reset Value | Description |
|---|---|---|---|---|
| 0xE000E100 | SETENA0 | R/W | 0 | Enable for external interrupt #0–31<br>bit[0] for interrupt #0 (exception #16)<br>bit[1] for interrupt #1 (exception #17)<br>…<br>bit[31] for interrupt #31 (exception #47)<br>Write 1 to set bit to 1; write 0 has no effect<br>Read value indicates the current status |
| 0xE000E104 | SETENA1 | R/W | 0 | Enable for external interrupt #32–63<br>Write 1 to set bit to 1; write 0 has no effect<br>Read value indicates the current status |
| 0xE000E108 | SETENA2 | R/W | 0 | Enable for external interrupt #64–95<br>Write 1 to set bit to 1; write 0 has no effect<br>Read value indicates the current status |
| … | - | - | - | - |
| 0xE000E180 | CLRENA0 | R/W | 0 | Clear enable for external interrupt #0–31<br>bit[0] for interrupt #0<br>bit[1] for interrupt #1<br>…<br>bit[31] for interrupt #31<br>Write 1 to clear bit to 0; write 0 has no effect<br>Read value indicates the current enable status |

- *Sur quelle adresse registre l'instruction suivante agit-elle ?*

  ```
  SETENA0 |= TIM1_UP_IRQChannel;
  ```

- *Quel est le numéro du canal d'interruption activé ? A quelle interruption correspond-il ?*

*Revenons au programme donné en début d'exercice :*

- *Que réalise le programme principal ?*
- *A quel moment la routine d'interruption sera-t-elle exécutée ?*
- *Que fait cette routine d'interruption ?*
- *Que se passera-t-il si l'on oublie de baisser le drapeau (UIF) ?*

## **Exercice 2 : Chenillard à double sens par** Interruptions du Timer 1 (TIM1)

*En se basant sur le principe du chenillard à double sens du TD2/TP2 et du programme précédent, compléter la routine d'interruption du programme ci dessous pour obtenir le même effet mais cette fois-ci sans la boucle infinie.*

```
/*  Includes  ------------------------------------------------------------------
*/
#include "stm32f10x.h"        // STM32F10x Library Definitions
//...//identique au programme ci-dessus
int LedPos = 0;                                    // position Led 0..7
int LedDir = 1;
//...//identique au programme ci-dessus
/* Configuration Timer 1 ------------------------------------------------------
------------*/
void cfgTimer1(void)
{
    RCC->APB2ENR |= (1 << 11);
    TIM1->PSC = 450;
    TIM1->ARR = 64000;
    TIM1->DIER |= UIE;
    TIM1->CR1 |= 0x0001;

    SETENA0 |= TIM1_UP_IRQChannel;
}

void TIM1_UP_TIM10_IRQHandler (void) ; //déclaration à compléter page suivant
int main(void)
{
  Enable_GPIO();
  Init_GPIO();
  cfgTimer1();

  while(1)
  { }
```

```
}
/* Traitement Interruption -----------------------------------------------------
--------------*/
void TIM1_UP_TIM10_IRQHandler (void)
{
    if(TIM1->SR & UIF)
    {
    // A compléter pour chenillard à double sens




      TIM1->SR &= ~UIF;
    }
}
```

## Fichier startup_stm32f10x_xl.s (extrait)

```
;******************** (C) COPYRIGHT 2011 STMicroelectronics ********************
;* File Name          : startup_stm32f10x_xl.s
;* Author             : MCD Application Team
;* Version            : V3.5.0
;* Date               : 11-March-2011
;* Description        : STM32F10x XL-Density Devices vector table for MDK-ARM
;*                      toolchain.
;*                      This module performs:
;*                      - Set the initial SP
;*                      - Set the initial PC == Reset_Handler
;*                      - Set the vector table entries with the exceptions ISR address
;*                      - Configure the clock system and also configure the external
;*                        SRAM mounted on STM3210E-EVAL board to be used as data
;*                        memory (optional, to be enabled by user)
;*                      - Branches to __main in the C library (which eventually
;*                        calls main()).
;*                      After Reset the CortexM3 processor is in Thread mode,
;*                      priority is Privileged, and the Stack is set to Main.
;* <<< Use Configuration Wizard in Context Menu >>>
;*******************************************************************************
; THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS
; WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME.
; AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT,
; INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE
; CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING
; INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
;*******************************************************************************

; Amount of memory (in bytes) allocated for Stack
; Tailor this value to your application needs
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp

; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x00000200

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit

                PRESERVE8
                THUMB


; Vector Table Mapped to Address 0 at Reset
                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size

__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     MemManage_Handler          ; MPU Fault Handler
                DCD     BusFault_Handler           ; Bus Fault Handler
                DCD     UsageFault_Handler         ; Usage Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     SVC_Handler                ; SVCall Handler
                DCD     DebugMon_Handler           ; Debug Monitor Handler
                DCD     0                          ; Reserved
```

```
        DCD     PendSV_Handler          ; PendSV Handler
        DCD     SysTick_Handler         ; SysTick Handler

        ; External Interrupts
        DCD     WWDG_IRQHandler            ; Window Watchdog
        DCD     PVD_IRQHandler             ; PVD through EXTI Line detect
        DCD     TAMPER_IRQHandler          ; Tamper
        DCD     RTC_IRQHandler             ; RTC
        DCD     FLASH_IRQHandler           ; Flash
        DCD     RCC_IRQHandler             ; RCC
        DCD     EXTI0_IRQHandler           ; EXTI Line 0
        DCD     EXTI1_IRQHandler           ; EXTI Line 1
        DCD     EXTI2_IRQHandler           ; EXTI Line 2
        DCD     EXTI3_IRQHandler           ; EXTI Line 3
        DCD     EXTI4_IRQHandler           ; EXTI Line 4
        DCD     DMA1_Channel1_IRQHandler   ; DMA1 Channel 1
        DCD     DMA1_Channel2_IRQHandler   ; DMA1 Channel 2
        DCD     DMA1_Channel3_IRQHandler   ; DMA1 Channel 3
        DCD     DMA1_Channel4_IRQHandler   ; DMA1 Channel 4
        DCD     DMA1_Channel5_IRQHandler   ; DMA1 Channel 5
        DCD     DMA1_Channel6_IRQHandler   ; DMA1 Channel 6
        DCD     DMA1_Channel7_IRQHandler   ; DMA1 Channel 7
        DCD     ADC1_2_IRQHandler          ; ADC1 & ADC2
        DCD     USB_HP_CAN1_TX_IRQHandler  ; USB High Priority or CAN1 TX
        DCD     USB_LP_CAN1_RX0_IRQHandler ; USB Low  Priority or CAN1 RX0
        DCD     CAN1_RX1_IRQHandler        ; CAN1 RX1
        DCD     CAN1_SCE_IRQHandler        ; CAN1 SCE
        DCD     EXTI9_5_IRQHandler         ; EXTI Line 9..5
        DCD     TIM1_BRK_TIM9_IRQHandler   ; TIM1 Break and TIM9
        DCD     TIM1_UP_TIM10_IRQHandler   ; TIM1 Update and TIM10
        DCD     TIM1_TRG_COM_TIM11_IRQHandler ; TIM1 Trigger and Commutation and TIM11
        DCD     TIM1_CC_IRQHandler         ; TIM1 Capture Compare
        DCD     TIM2_IRQHandler            ; TIM2
        DCD     TIM3_IRQHandler            ; TIM3
        DCD     TIM4_IRQHandler            ; TIM4
        DCD     I2C1_EV_IRQHandler         ; I2C1 Event
        DCD     I2C1_ER_IRQHandler         ; I2C1 Error
        DCD     I2C2_EV_IRQHandler         ; I2C2 Event
        DCD     I2C2_ER_IRQHandler         ; I2C2 Error
        DCD     SPI1_IRQHandler            ; SPI1
        DCD     SPI2_IRQHandler            ; SPI2
        DCD     USART1_IRQHandler          ; USART1
        DCD     USART2_IRQHandler          ; USART2
        DCD     USART3_IRQHandler          ; USART3
        DCD     EXTI15_10_IRQHandler       ; EXTI Line 15..10
        DCD     RTCAlarm_IRQHandler        ; RTC Alarm through EXTI Line
        DCD     USBWakeUp_IRQHandler       ; USB Wakeup from suspend
        DCD     TIM8_BRK_TIM12_IRQHandler  ; TIM8 Break and TIM12
        DCD     TIM8_UP_TIM13_IRQHandler   ; TIM8 Update and TIM13
        DCD     TIM8_TRG_COM_TIM14_IRQHandler ; TIM8 Trigger and Commutation and TIM14
        DCD     TIM8_CC_IRQHandler         ; TIM8 Capture Compare
        DCD     ADC3_IRQHandler            ; ADC3
        DCD     FSMC_IRQHandler            ; FSMC
        DCD     SDIO_IRQHandler            ; SDIO
        DCD     TIM5_IRQHandler            ; TIM5
        DCD     SPI3_IRQHandler            ; SPI3
        DCD     UART4_IRQHandler           ; UART4
        DCD     UART5_IRQHandler           ; UART5
        DCD     TIM6_IRQHandler            ; TIM6
        DCD     TIM7_IRQHandler            ; TIM7
        DCD     DMA2_Channel1_IRQHandler   ; DMA2 Channel1
        DCD     DMA2_Channel2_IRQHandler   ; DMA2 Channel2
        DCD     DMA2_Channel3_IRQHandler   ; DMA2 Channel3
        DCD     DMA2_Channel4_5_IRQHandler ; DMA2 Channel4 & Channel5
__Vectors_End
...........
```

## 8.4.3 External interrupt configuration register 1 (AFIO_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| EXTI3[3:0] | | | | EXTI2[3:0] | | | | EXTI1[3:0] | | | | EXTI0[3:0] | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]:** EXTI x configuration (x= 0 to 3)
These bits are written by software to select the source input for EXTIx external interrupt.
Refer to *Section 9.2.5: External interrupt/event line mapping on page 176*
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

## 8.4.5 External interrupt configuration register 3 (AFIO_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| EXTI11[3:0] | | | | EXTI10[3:0] | | | | EXTI9[3:0] | | | | EXTI8[3:0] | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x= 8 to 11)
These bits are written by software to select the source input for EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

## 9.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | MR19 | MR18 | MR17 | MR16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| MR15 | MR14 | MR13 | MR12 | MR11 | MR10 | MR9 | MR8 | MR7 | MR6 | MR5 | MR4 | MR3 | MR2 | MR1 | MR0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:20 Reserved, must be kept at reset value (0).

Bits 19:0 **MRx:** Interrupt Mask on line x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

*Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.*

### 9.3.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | TR19 | TR18 | TR17 | TR16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| TR15 | TR14 | TR13 | TR12 | TR11 | TR10 | TR9 | TR8 | TR7 | TR6 | TR5 | TR4 | TR3 | TR2 | TR1 | TR0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:20 Reserved, must be kept at reset value (0).

Bits 19:0 **TRx:** Rising trigger event configuration bit of line x
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.
*Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.*

*Note: The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a rising edge on external interrupt line occurs during writing of EXTI_RTSR register, the pending bit will not be set.*

*Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.*

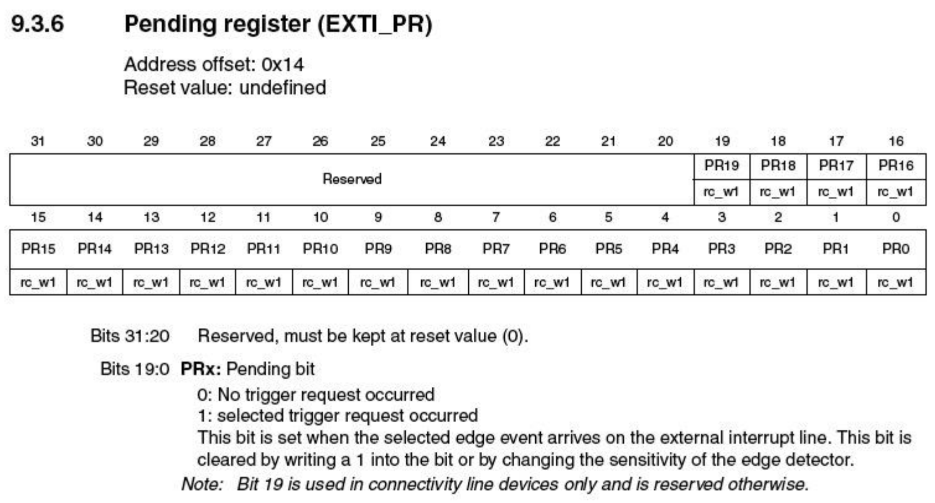
### 9.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | TR19 | TR18 | TR17 | TR16 |
| | | | | | | | | | | | | rw | rw | rw | rw |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| TR15 | TR14 | TR13 | TR12 | TR11 | TR10 | TR9 | TR8 | TR7 | TR6 | TR5 | TR4 | TR3 | TR2 | TR1 | TR0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:20 Reserved, must be kept at reset value (0).

Bits 19:0 **TRx:** Falling trigger event configuration bit of line x
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.
*Note: Bit 19 used in connectivity line devices and is reserved otherwise.*



### 9.3.6 Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PR19 | PR18 | PR17 | PR16 |
| | | | | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PR15 | PR14 | PR13 | PR12 | PR11 | PR10 | PR9 | PR8 | PR7 | PR6 | PR5 | PR4 | PR3 | PR2 | PR1 | PR0 |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

Bits 31:20 Reserved, must be kept at reset value (0).

Bits 19:0 **PRx:** Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.
*Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.*