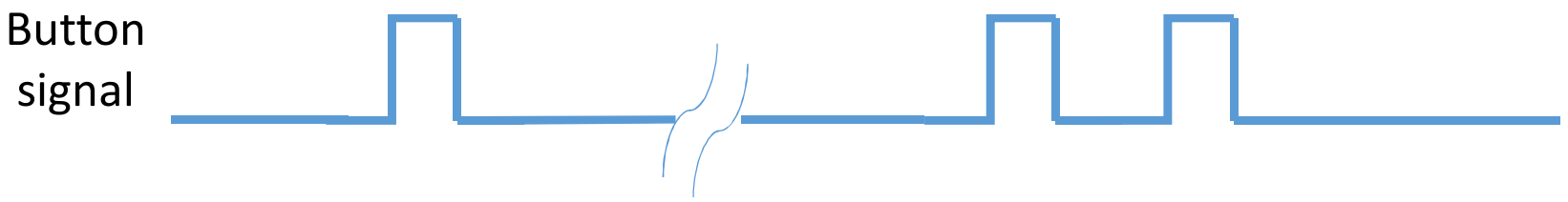
Lesson 4
Interrupt

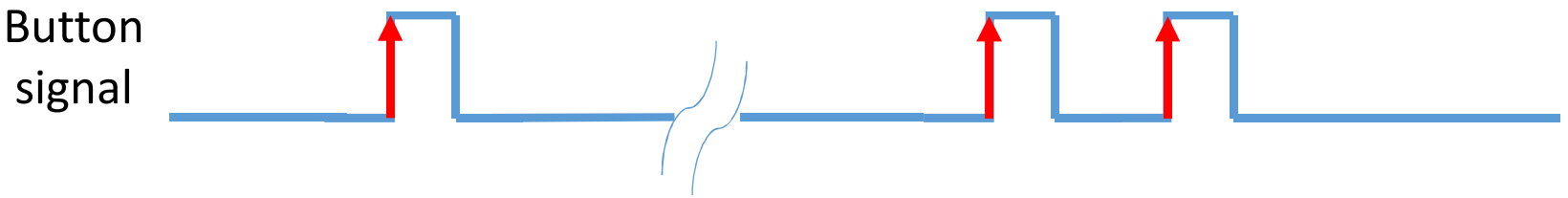
Lecturer: Harvard Tseng

When will user press button?



We don’t want CPU polling, wasting time to wait, these unexpected signal.

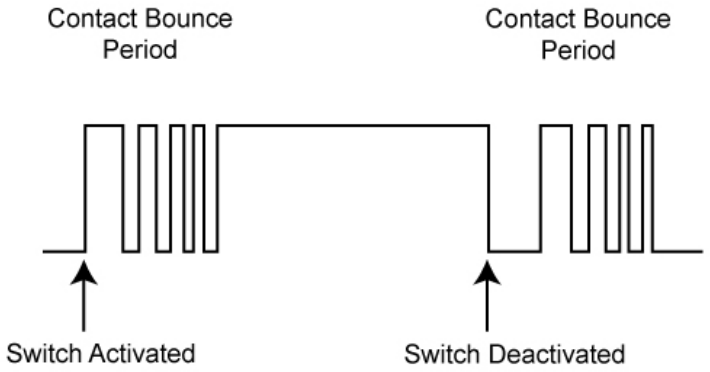
Do what?

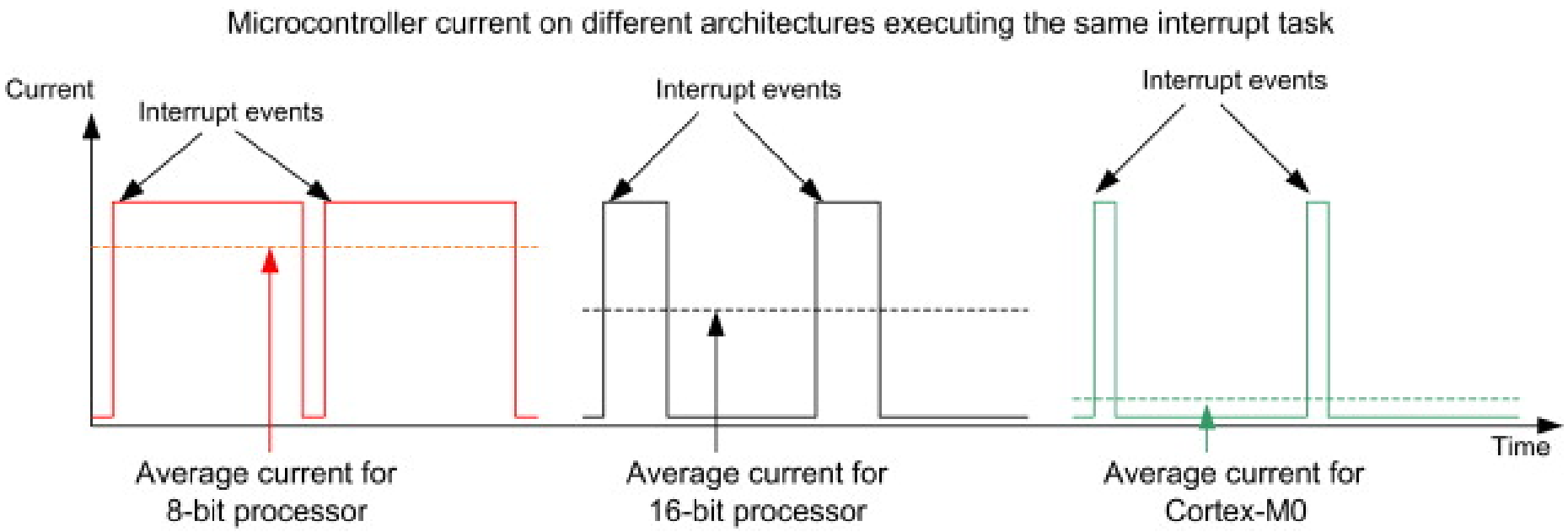


Detect rising/falling edge, then interrupt the CPU to handle exception.

Can interrupt solve switch bouncing problem?

Maybe 8051 can,
but not Cortex-M.

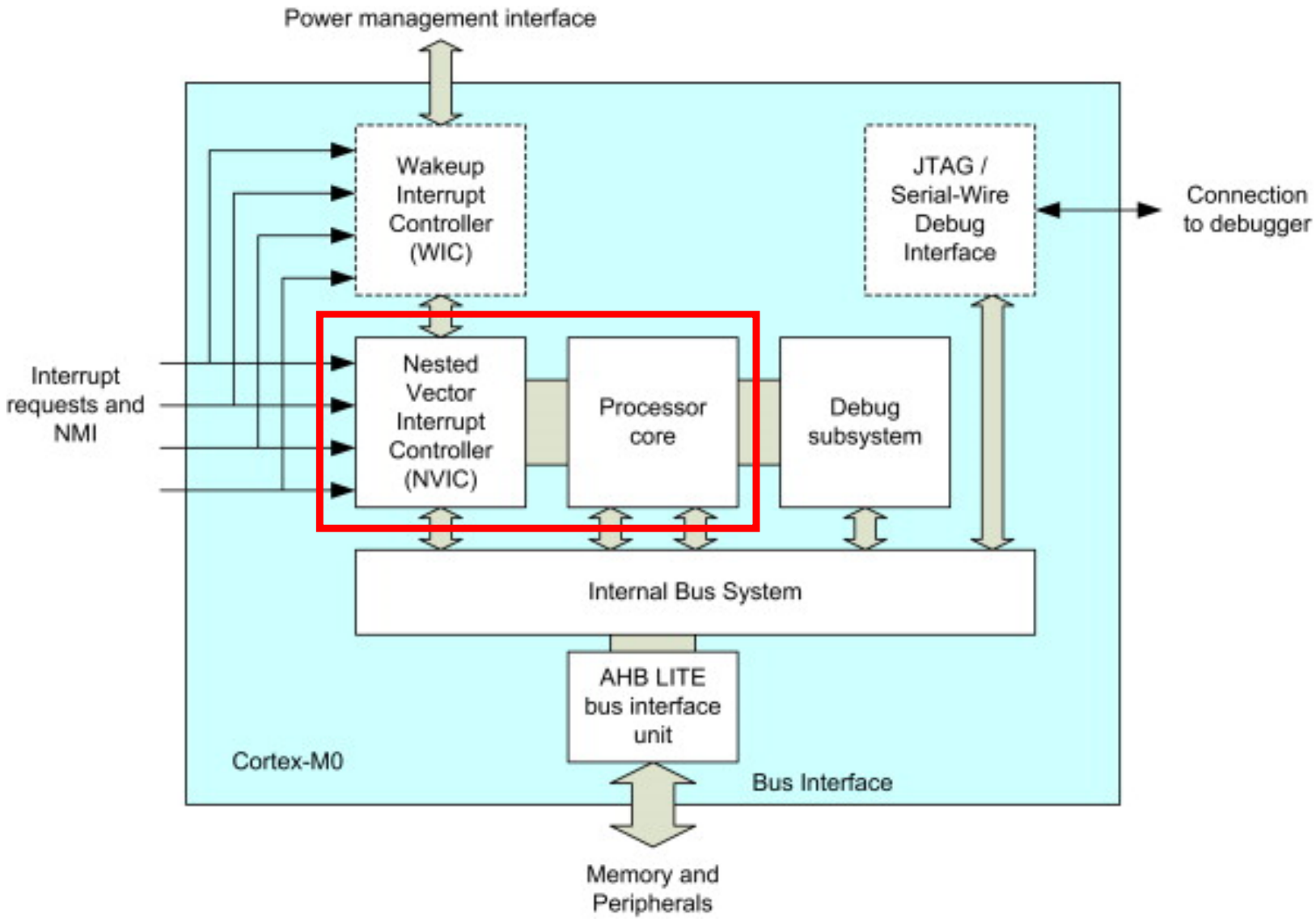




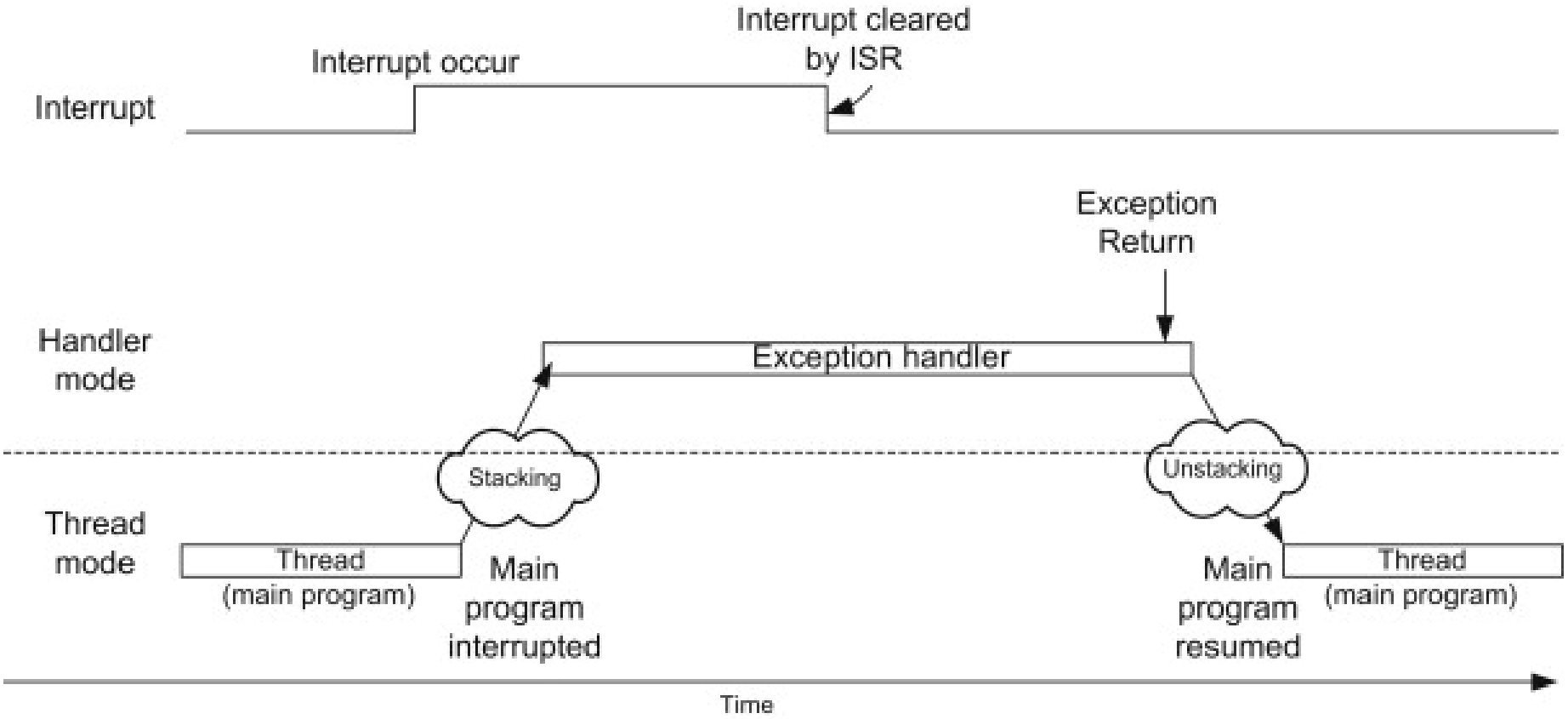
Background

Let’s look inside the core.

Simplified Block Diagram

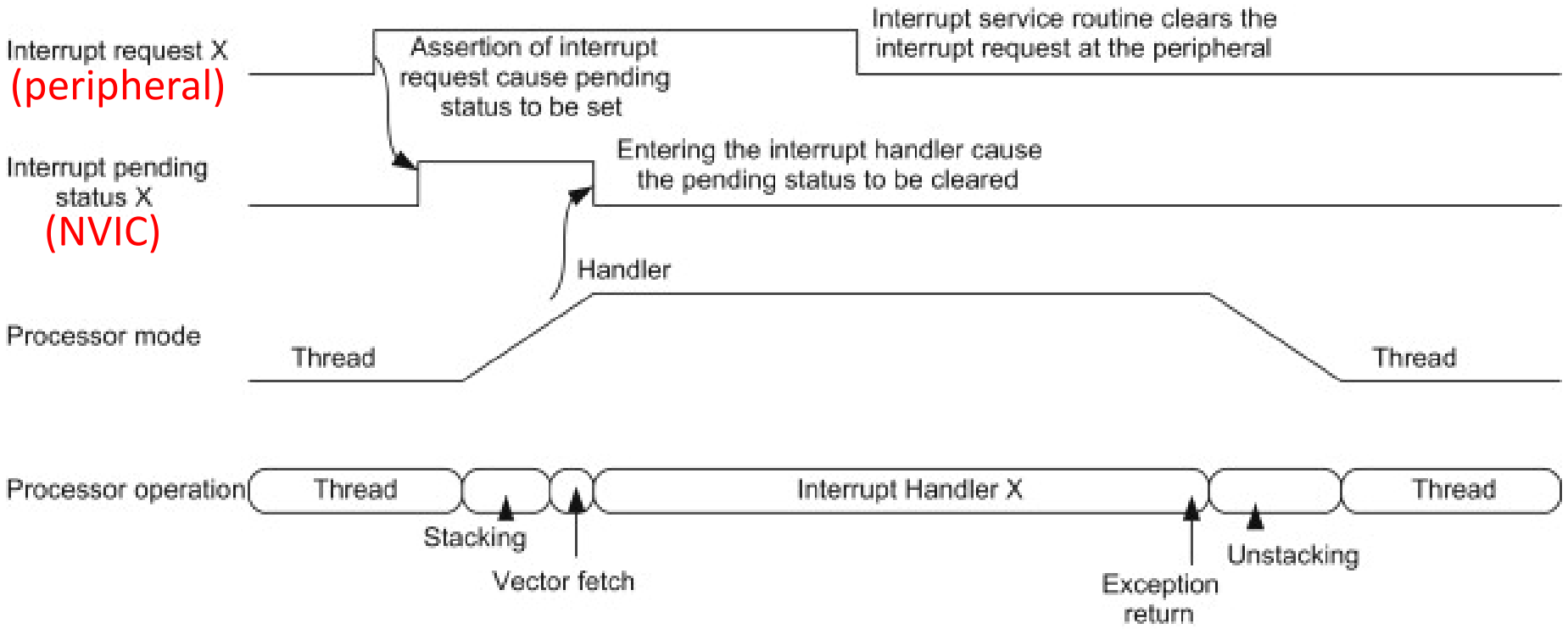


Simply view at exception entry & exit



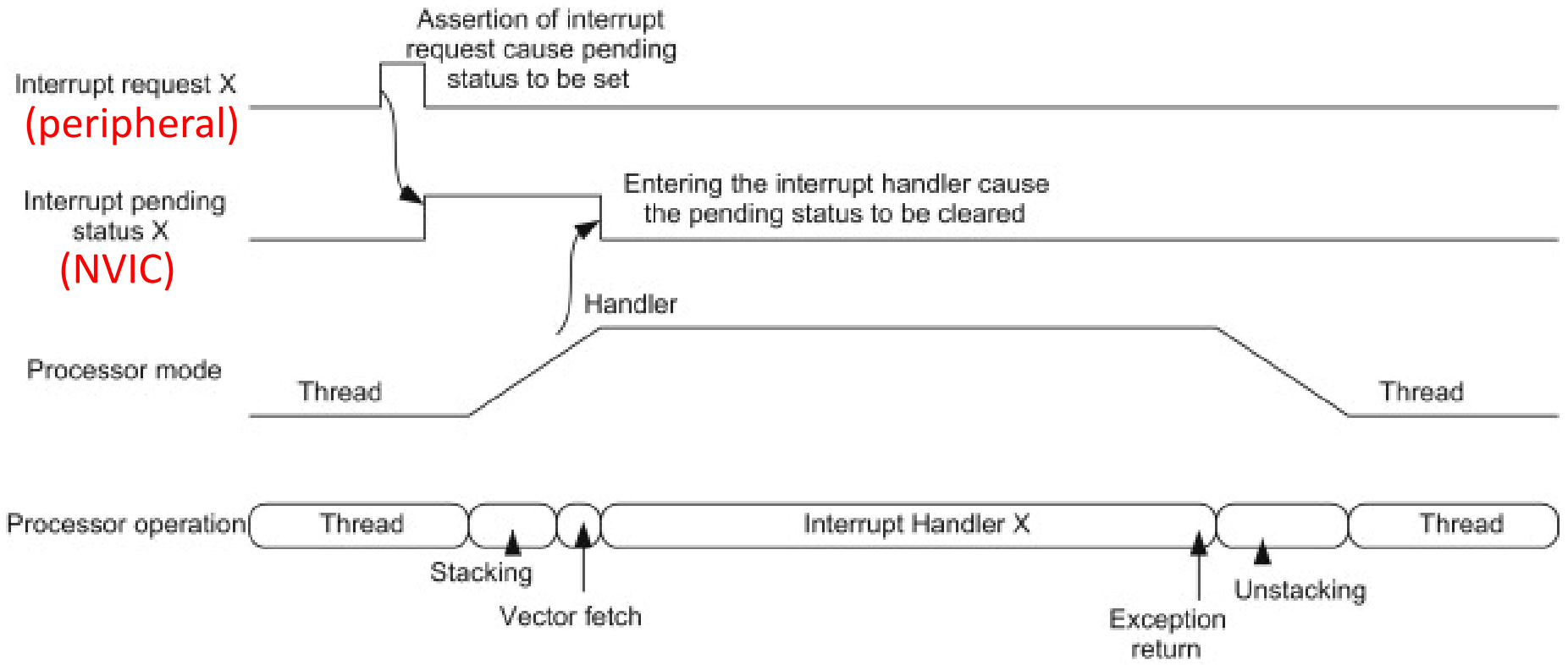
Level Trigger

- Peripheral interrupt request will store in status register(TIMx) or pending register(EXTI).



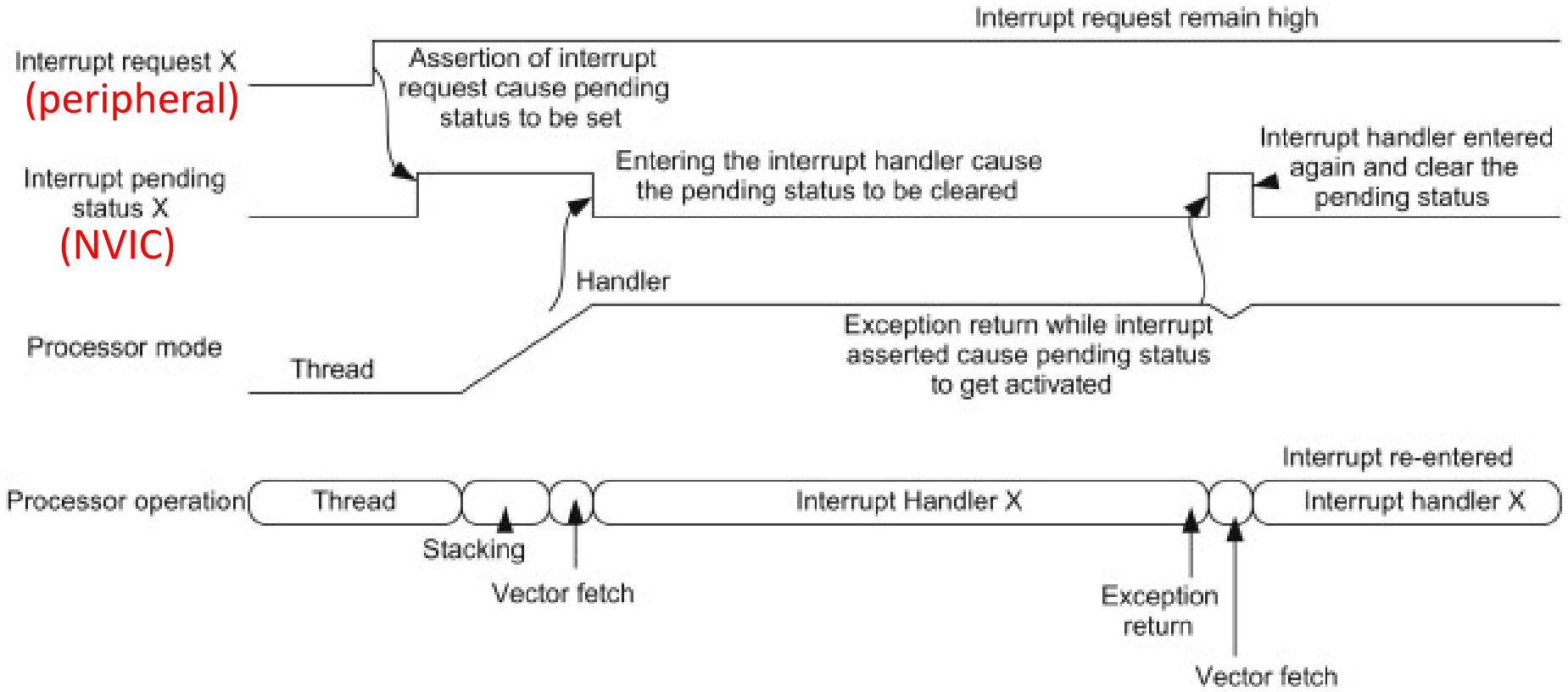
Pulse Trigger

- Pending status register will hold the request until the interrupt is being served.



If do not clear interrupt request

- Will re-enter interrupt handler again and again.



SYSCFG

System configuration controller

Main purposes

- Enabling/disabling I2C Fast Mode Plus on some IO ports
- Remapping some DMA trigger sources to different DMA channels
- Remapping the memory located at the beginning of the code area
- Managing the external interrupt line connection to the GPIOs
- …

Register

10.1.2 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration bits (x = 0 to 3)

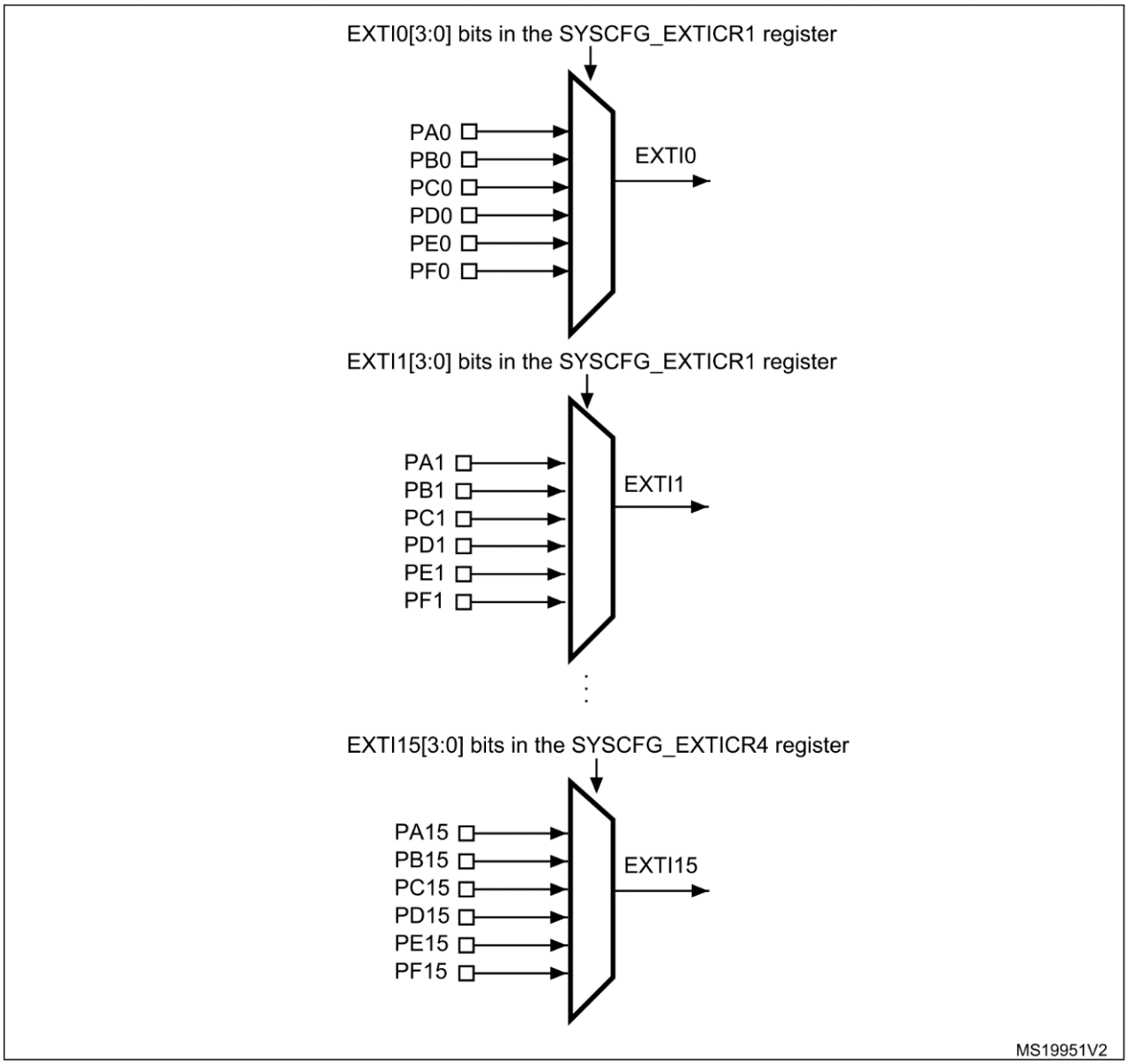
These bits are written by software to select the source input for the EXTIx external interrupt.

x000: PA[x] pin
x001: PB[x] pin
x010: PC[x] pin
x011: PD[x] pin
x100: PE[x] pin
x101: PF[x] pin
other configurations: reserved

EXTI

Extended interrupts and events controller

External interrupt/event GPIO mapping



Register

- EXTI_IMR – Interrupt mask register – to mask interrupt request or not.
- EXTI_RTSR - Rising trigger selection register
- EXTI_FTSR - Falling trigger selection register
- EXTI_PR - Pending register - is set when the selected edge event arrives on the external interrupt line.

EXTI interrupt selection

- Eg. PC13 as interrupt source.
- RCC_APB2ENR |= RCC_APB2ENR_SYSCFGEN; // Enable SYSCFG clock
- SYSCFG_EXTICR4 |= SYSCFG_EXTICR4_EXTI13 & SYSCFG_EXTICR4_EXTI13_PC; // Select the source input for EXTI13
- EXTI_IMR |= EXTI_IMR_MR13; // Set mask bit(EXTI_Line13)
- EXTI_RTSR |= EXTI_RTSR_TR13; // Active when rising edge occur
- NVIC_EnableIRQ(EXTI15_10_IRQn); // Enable NVIC IRQ channel

Peripheral Interrupt Handler

- Every interrupt handler's name is already well defined in startup_<device>.s .
- Implement handler function under main() or in independent source file.

```
int main(void){
        …
}
void EXTI15_10_IRQHandler(void){
        if((EXTI->PR & EXTI_PR_PR13) != 0){
            …
            EXTI->PR = EXTI_PR_PR13;
        }
}
```

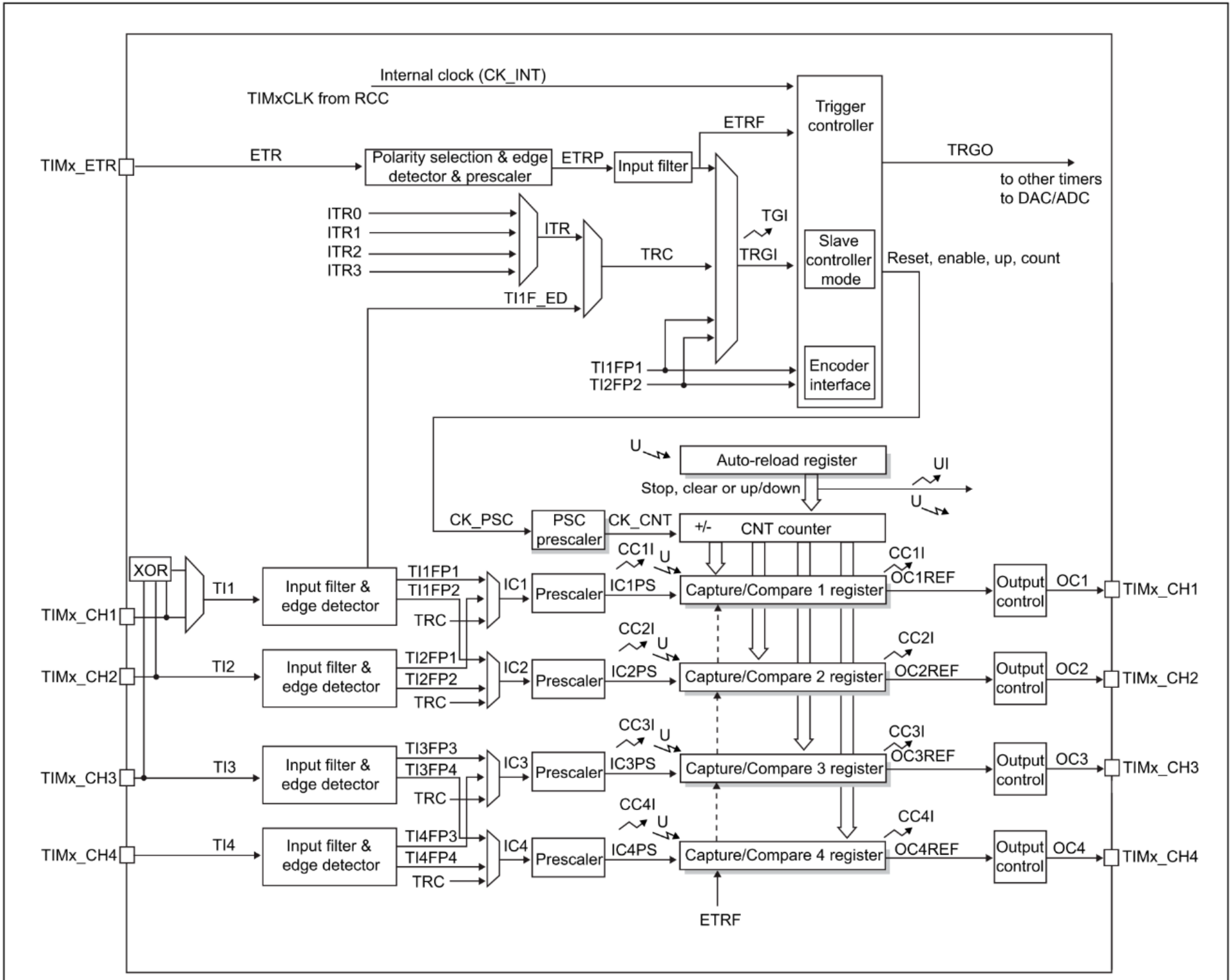
Exercise 1

- If button pressed, increase frequency by 1. When frequency is bigger than 10, set to 1.
- Using ISR to handle actions when button pressed.
- Using the code from last lesson to generate periodic blink(polling).

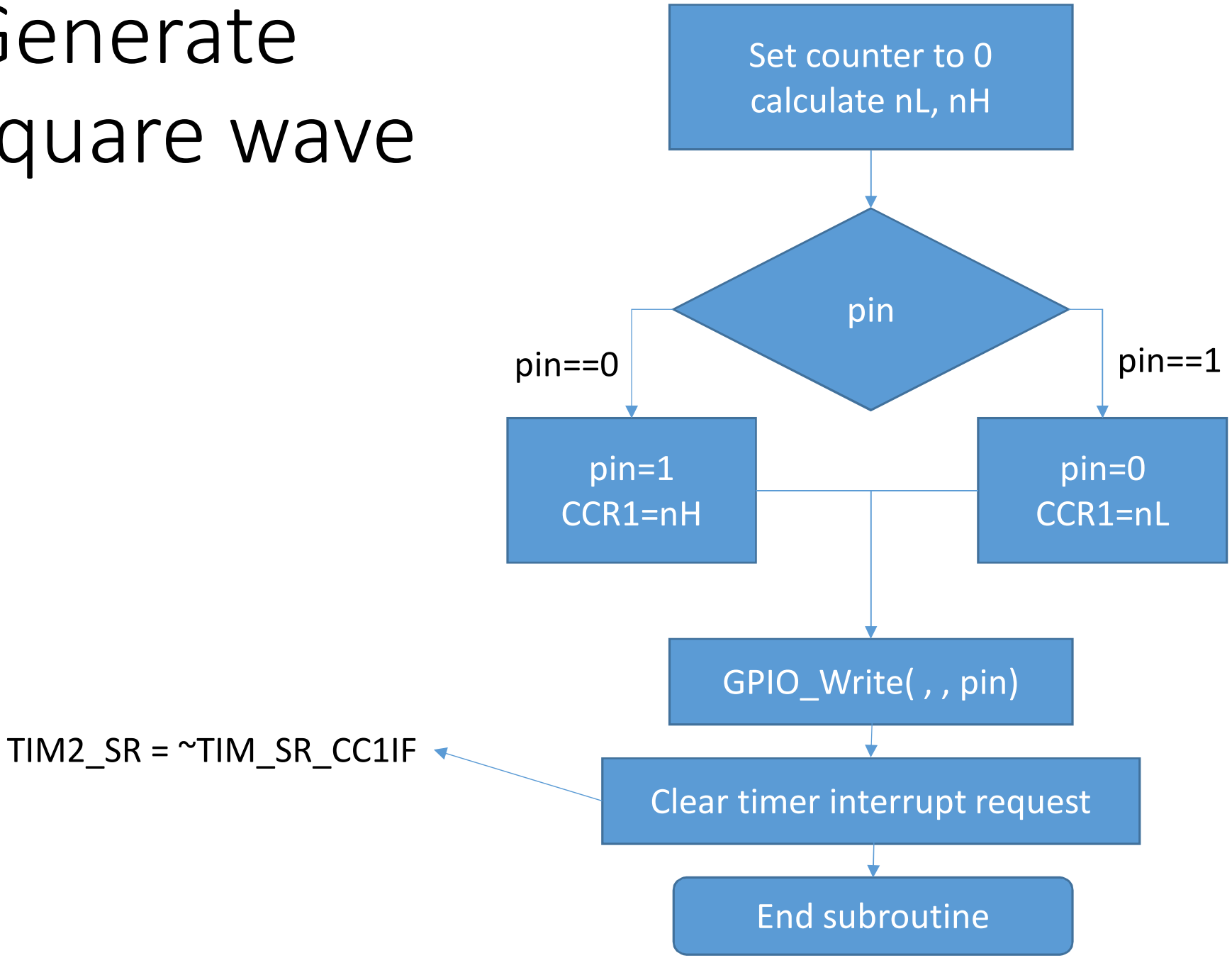
Timer Interrupt

Register

- TIMx_CCRy - capture/compare register – if counter matches CCR value, will generate interrupt.
- TIMx_SR – status register – record CCR interrupt information.
- TIMx_DIER - DMA/Interrupt enable register – enable DMA/Interrupt request.


Internal clock (CK_INT)
TIMxCLK from RCC
Trigger controller
ETRF
TRGO
to other timers
to DAC/ADC
TIMx_ETR
ETR
Polarity selection & edge detector & prescaler
ETRP
Input filter
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave controller mode
Reset, enable, up, count
TI1F_ED
TI1FP1
TI2FP2
Encoder interface
U
Auto-reload register
UI
Stop, clear or up/down
CK_PSC
PSC prescaler
CK_CNT
+/-
CNT counter
XOR
TI1
TI2
TI3
TI4
Input filter & edge detector
TI1FP1
TI1FP2
TI2FP1
TI2FP2
TI3FP3
TI3FP4
TI4FP3
TI4FP4
IC1
IC2
IC3
IC4
Prescaler
IC1PS
IC2PS
IC3PS
IC4PS
CC1I
CC2I
CC3I
CC4I
Capture/Compare 1 register
Capture/Compare 2 register
Capture/Compare 3 register
Capture/Compare 4 register
OC1REF
OC2REF
OC3REF
OC4REF
Output control
OC1
OC2
OC3
OC4
TIMx_CH1
TIMx_CH2
TIMx_CH3
TIMx_CH4

Generate square wave



Timer CCR interrupt configuration

- TIM2_DIER |= TIM_DIER_CC1IE; // Capture/Compare 1 interrupt enable
- TIM2_CCR1 = 100; // Give a match value to compare
- NVIC_EnableIRQ(TIM2_IRQn); // Enable NVIC IRQ channel

Exercise 2

- Following from Exercise 1. No more polling!
- Using ISR to handle actions when button pressed and counter matches compare value.
- Your main function should run nothing!