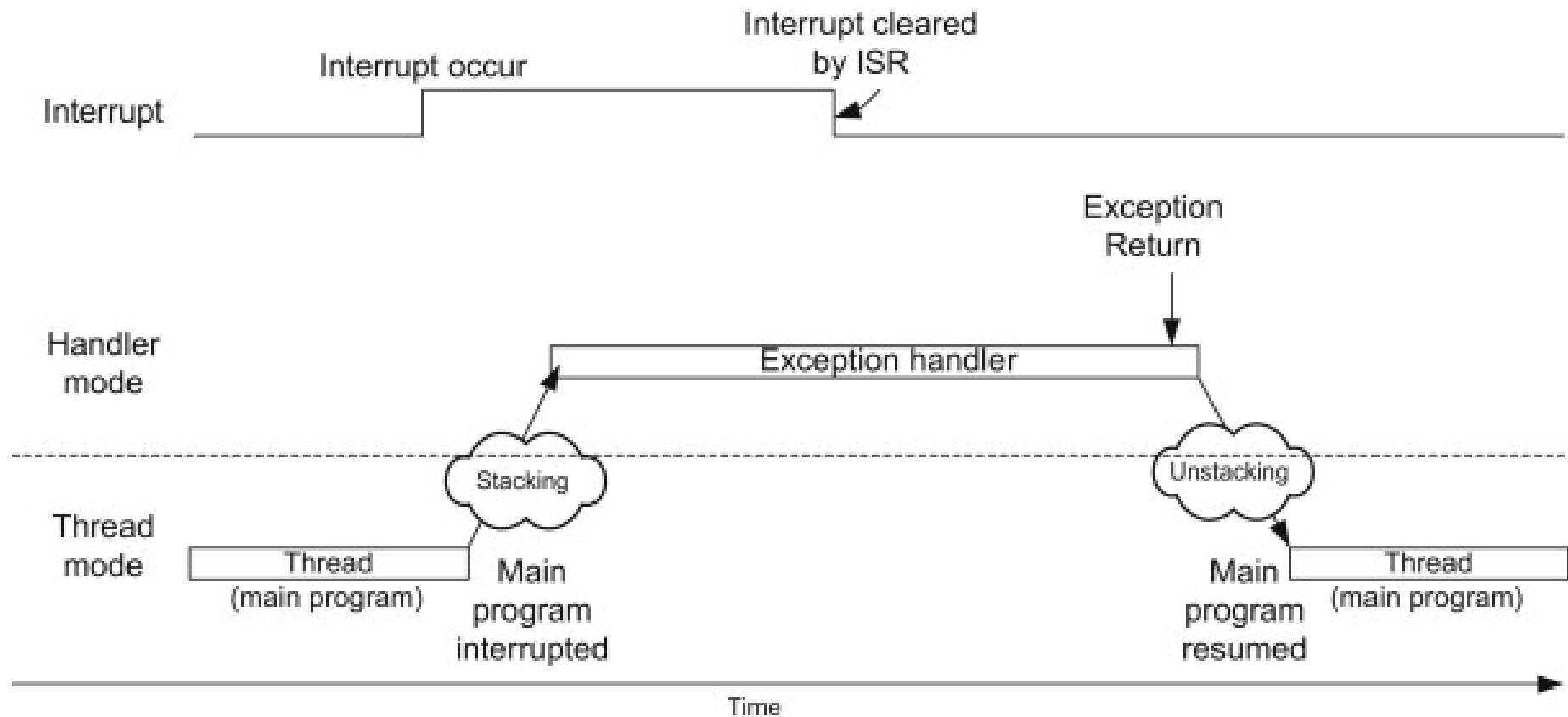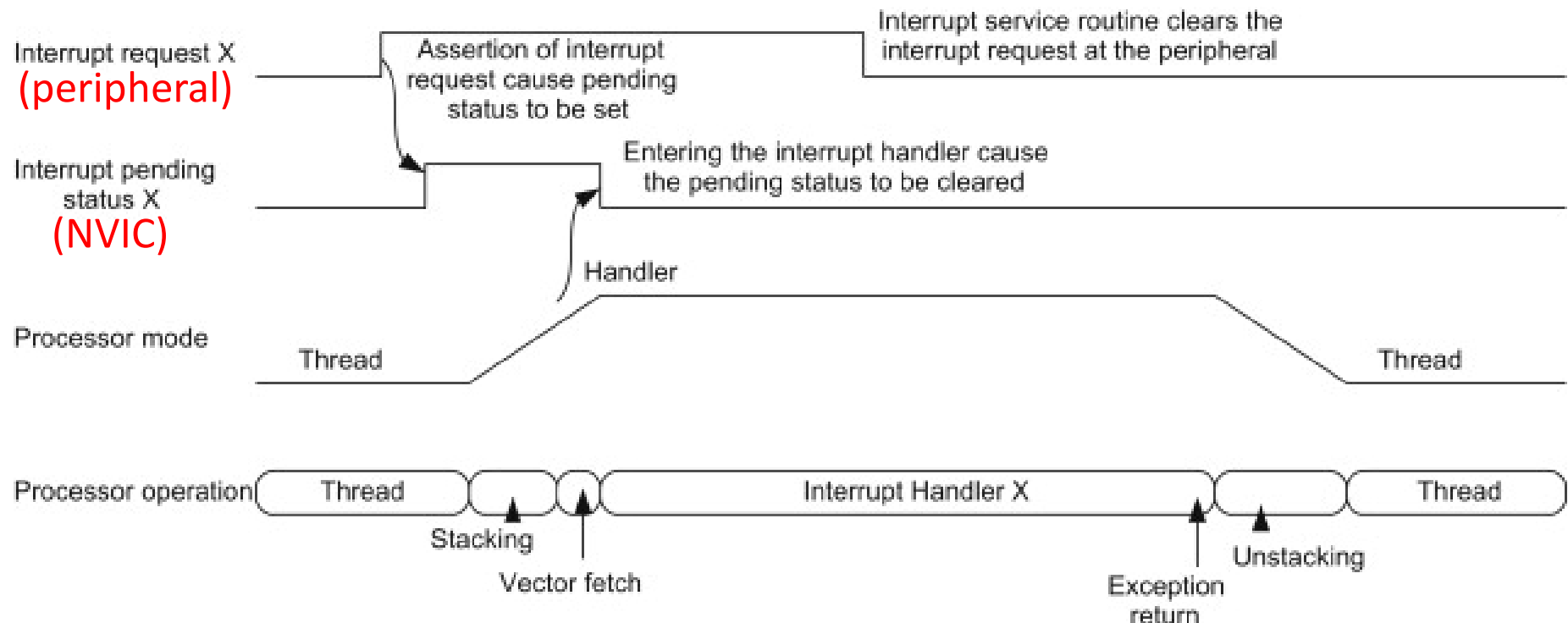# Lesson 4
Interrupt

Lecturer: Harvard Tseng

Simply view at exception entry & exit

Level Trigger

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

Interrupt request X
(peripheral)

Assertion of interrupt request cause pending status to be set

Interrupt service routine clears the interrupt request at the peripheral

Interrupt pending status X
(NVIC)

Entering the interrupt handler cause the pending status to be cleared

Handler

Processor mode

Thread

Thread

Processor operation

Thread

Interrupt Handler X

Thread

Stacking

Vector fetch

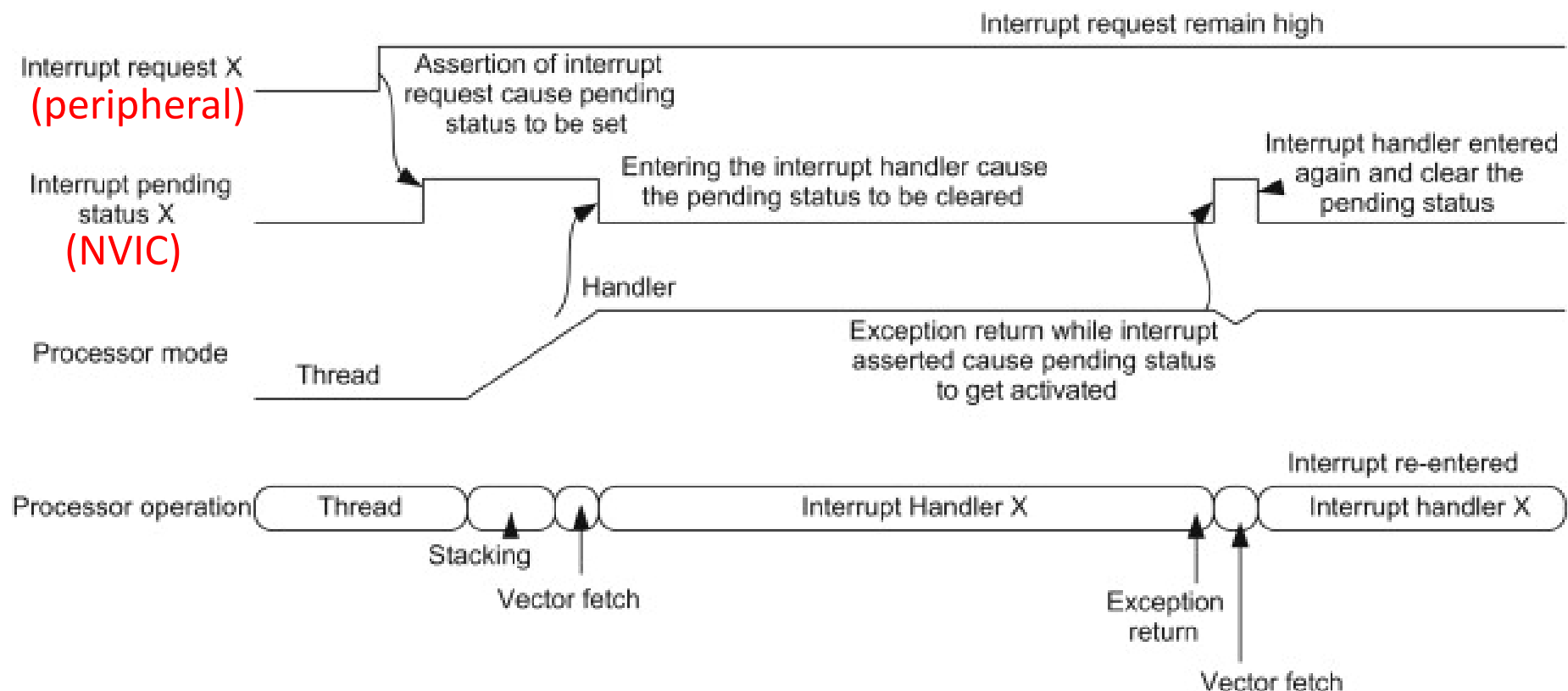Exception return

Unstacking

Pulse Trigger

- Pending status register will hold the request until the interrupt is being served.
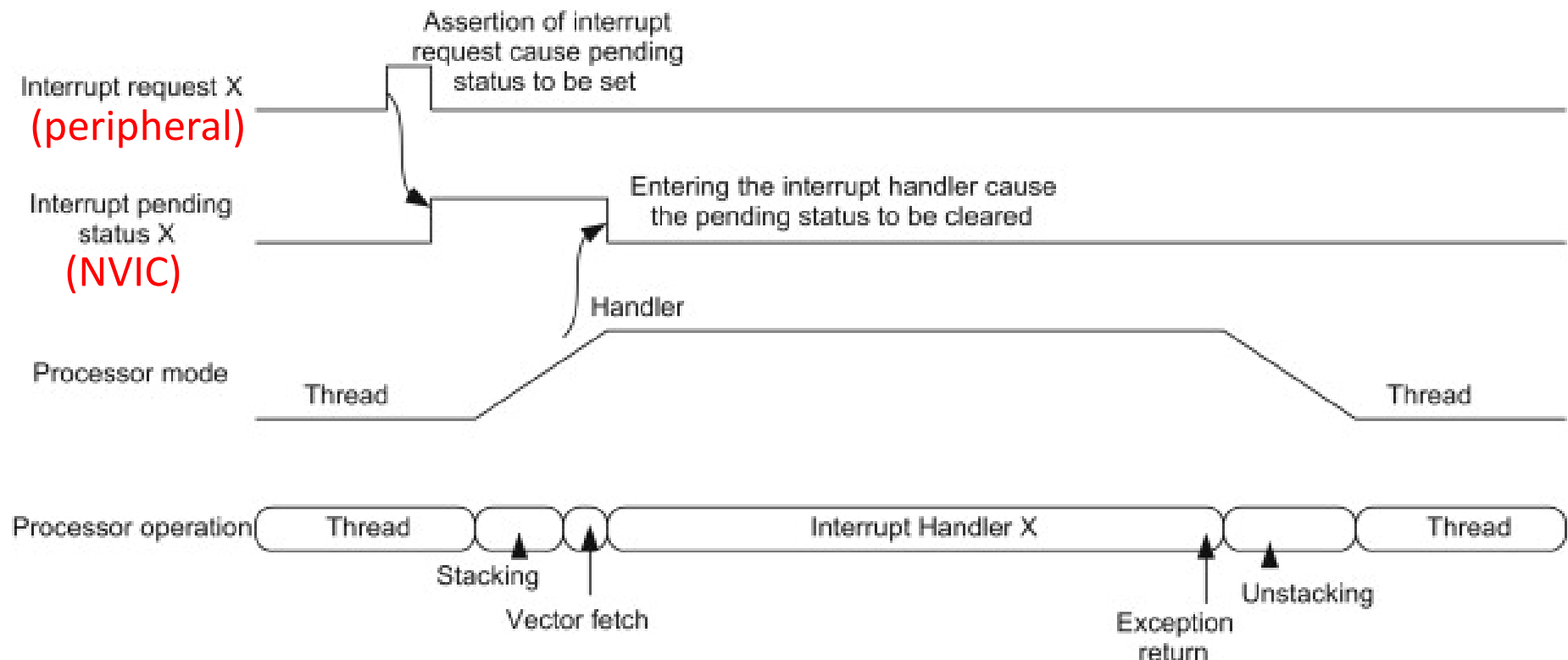
If do not clear interrupt request

- 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
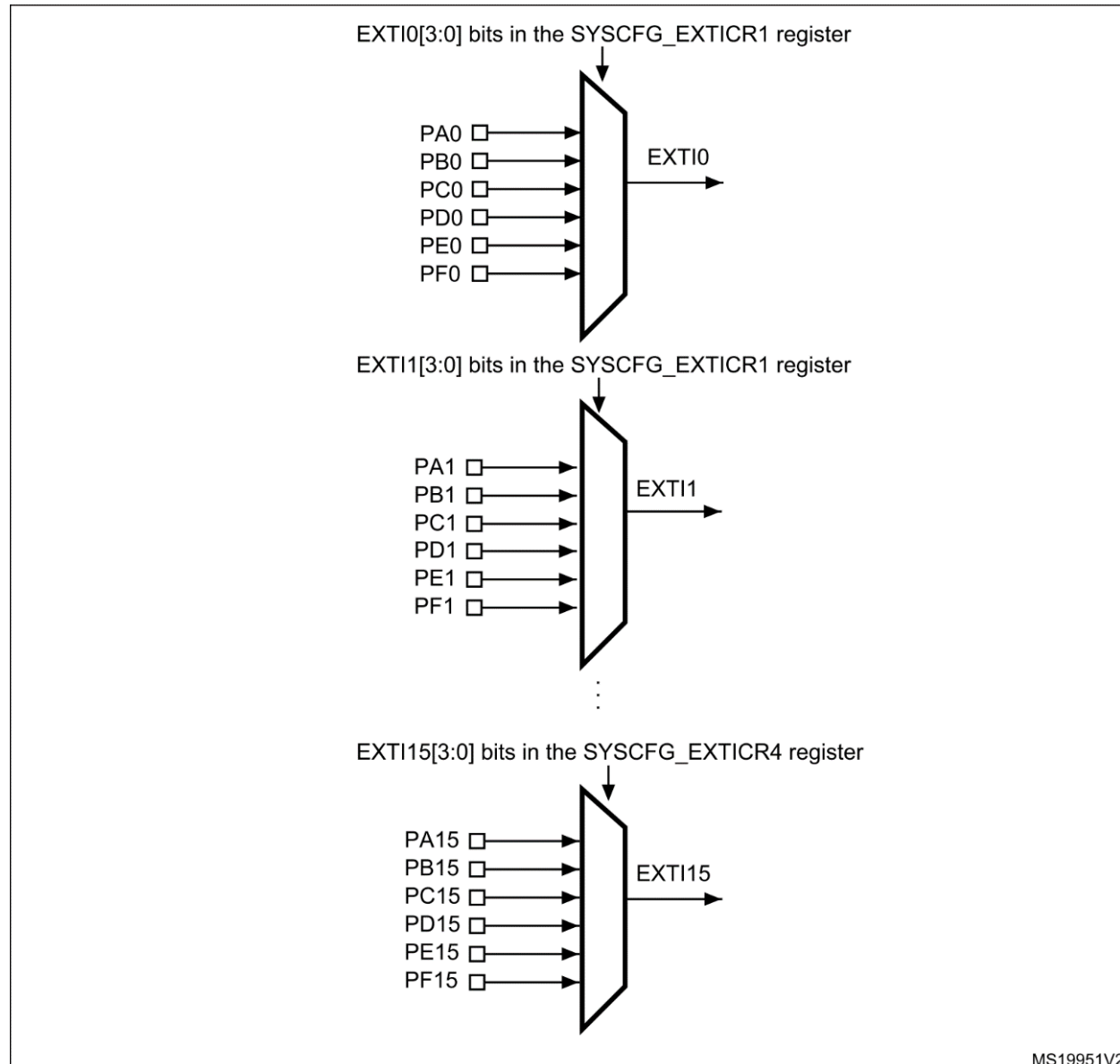x010: PC[x] pin
x011: PD[x] pin
x100: PE[x] pin
x101: PF[x] pin
other configurations: reserved

EXTI

Extended interrupts and events controller

External interrupt/event GPIO mapping

EXTI0[3:0] bits in the SYSCFG_EXTICR1 register

PA0
PB0
PC0   EXTI0
PD0
PE0
PF0

EXTI1[3:0] bits in the SYSCFG_EXTICR1 register

PA1
PB1
PC1   EXTI1
PD1
PE1
PF1

EXTI15[3:0] bits in the SYSCFG_EXTICR4 register

PA15
PB15
PC15   EXTI15
PD15
PE15
PF15

MS19951V2

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.
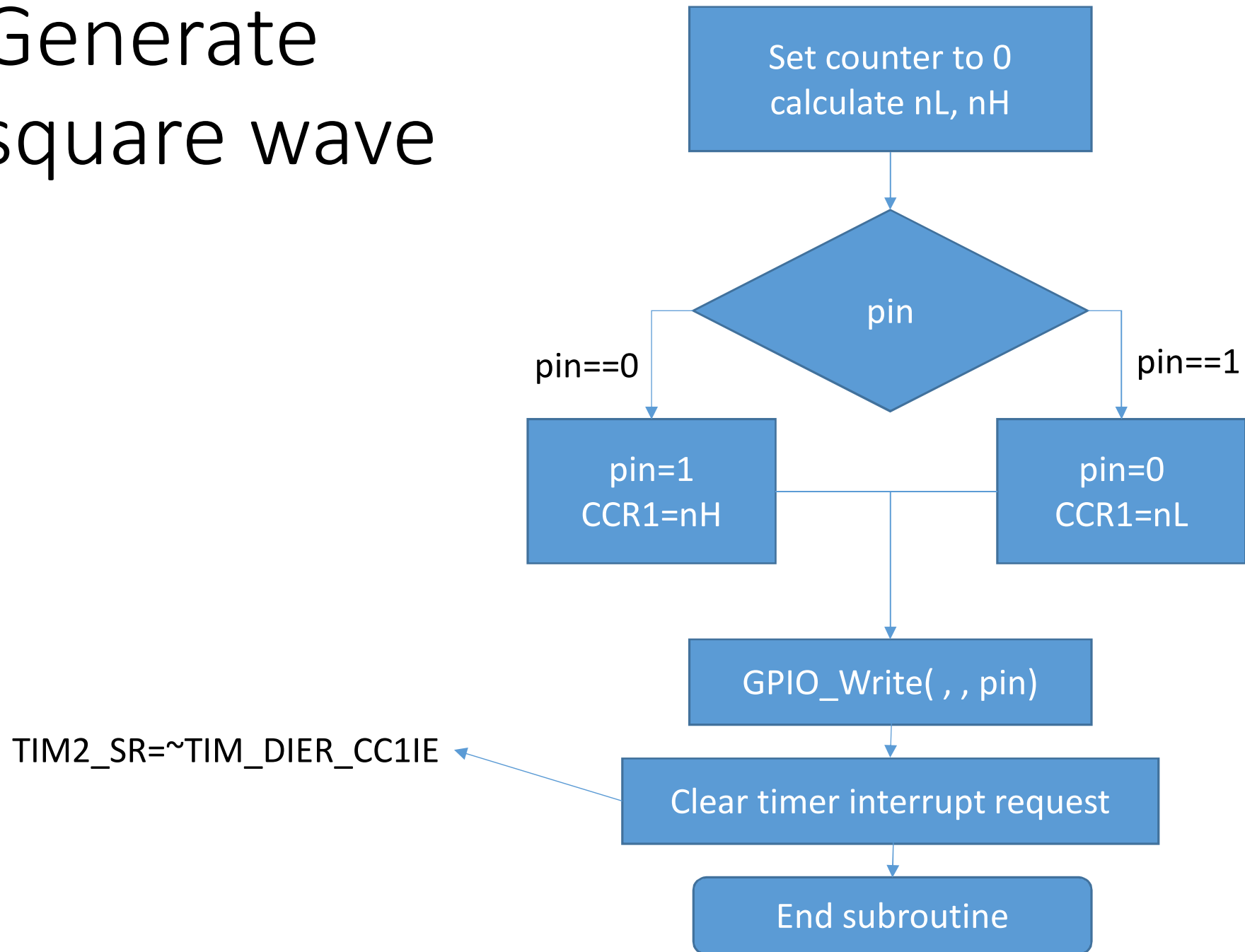
EXTI interrupt selection

- ## Eg. PA0 as interrupt source.

- RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN; // Enable SYSCFG clock

- SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0 & SYSCFG_EXTICR1_EXTI0_PA;   // Select the source input for EXTI0

- EXTI_IMR |= EXTI_IMR_IM0;   // Set mask bit(EXTI_Line0)

- EXTI_RTSR |= EXTI_RTSR_RT0;   // Active when rising edge occur

- NVIC_EnableIRQ(EXTI0_1_IRQn); // Enable NVIC IRQ channel

Timer

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.

Generate square wave

Set counter to 0
calculate nL, nH

pin

pin==0

pin==1

pin=1
CCR1=nH

pin=0
CCR1=nL

GPIO_Write(, , pin)

TIM2_SR=~TIM_DIER_CC1IE

Clear timer interrupt request

End subroutine

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

Peripheral Interrupt Handler

- Every interrupt handle's name is already well defined in startup_<device>.s .

- Implement handler function under main() or in independent source file.

```c
int main(void){

    …

}
void TIM2_IRQHandler(void){

    …

}
```

Exercise

- If button pressed, increase frequency by 1. When frequency is bigger than 10, set to 1.

- Use ISR to handle actions when button pressed or timer match compare value.