BME3321:Introduction to Microcontroller Programming

Topic 5: Interrupt Management in ARM Cortex MCUs

Asst. Prof. Dr. İsmail Cantürk

The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors (ch 8)

STM32F407 reference manual (ch 12)

Also check Mastering STM32 (ch7)

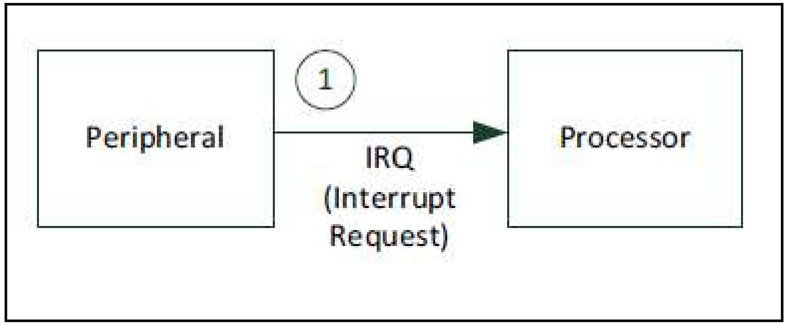
What are Interrupts?

- In most microcontrollers, the interrupt feature enables a peripheral or an external hardware to send a request to a processor so that the processor can execute a piece of code to service the request.

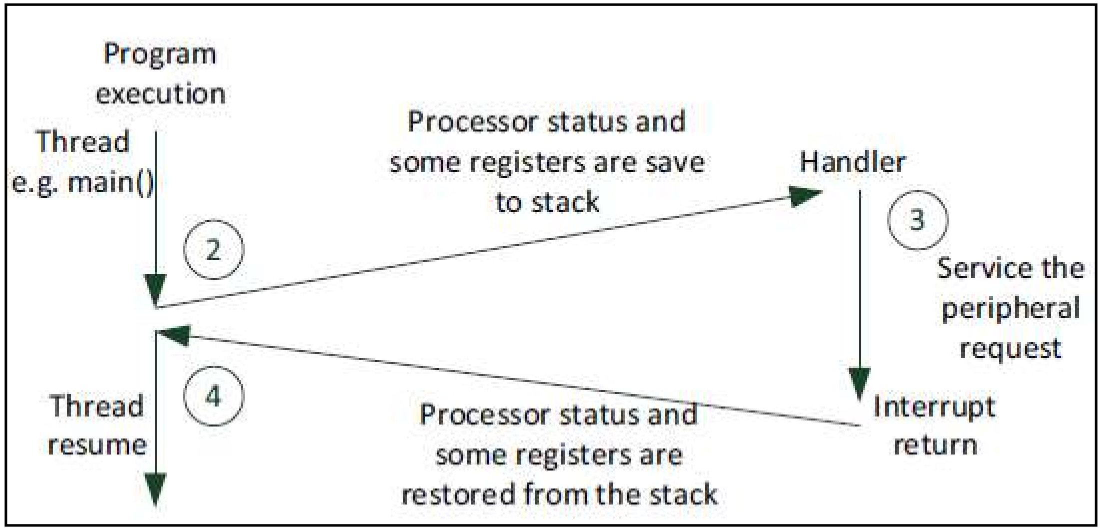
- The process involves suspending the current executing task, or wake up from sleep mode, and execute the piece of software code called interrupt handler or interrupt service routine to service the request.

- After the request is serviced, the processor can then resume the previous interrupted code.

Interrupts



- A peripheral generates an interrupt request (IRQ) to the processor.
- The processor detects and accepts the IRQ.
- Current task is suspended, push operation to stack memory takes place.
- The processor locates the starting address of the interrupt handler from the vector table, and then executes the interrupt handler associated with this IRQ.
- The processor finishes the handler execution, restores the information previously pushed to the stack (popping), and resumes the interrupted task.



Some terminologies about interrupt

Interrupt Requests (IRQs): generated by peripherals (e.g., timers) including external interrupt inputs via GPIO pins.

Non-Maskable Interrupt (NMI): A special IRQ with the highest priority level and cannot be disabled. Typically generated by peripherals like the watchdog timer.

Interrupt handlers: The software code that gets executed when an interrupt occurred is called interrupt handler or Interrupt Service Routine (ISR).

Nested Vectored Interrupt Controller (NVIC): A programmable hardware unit inside the Cortex-M processors to handle the management of interrupts requests.

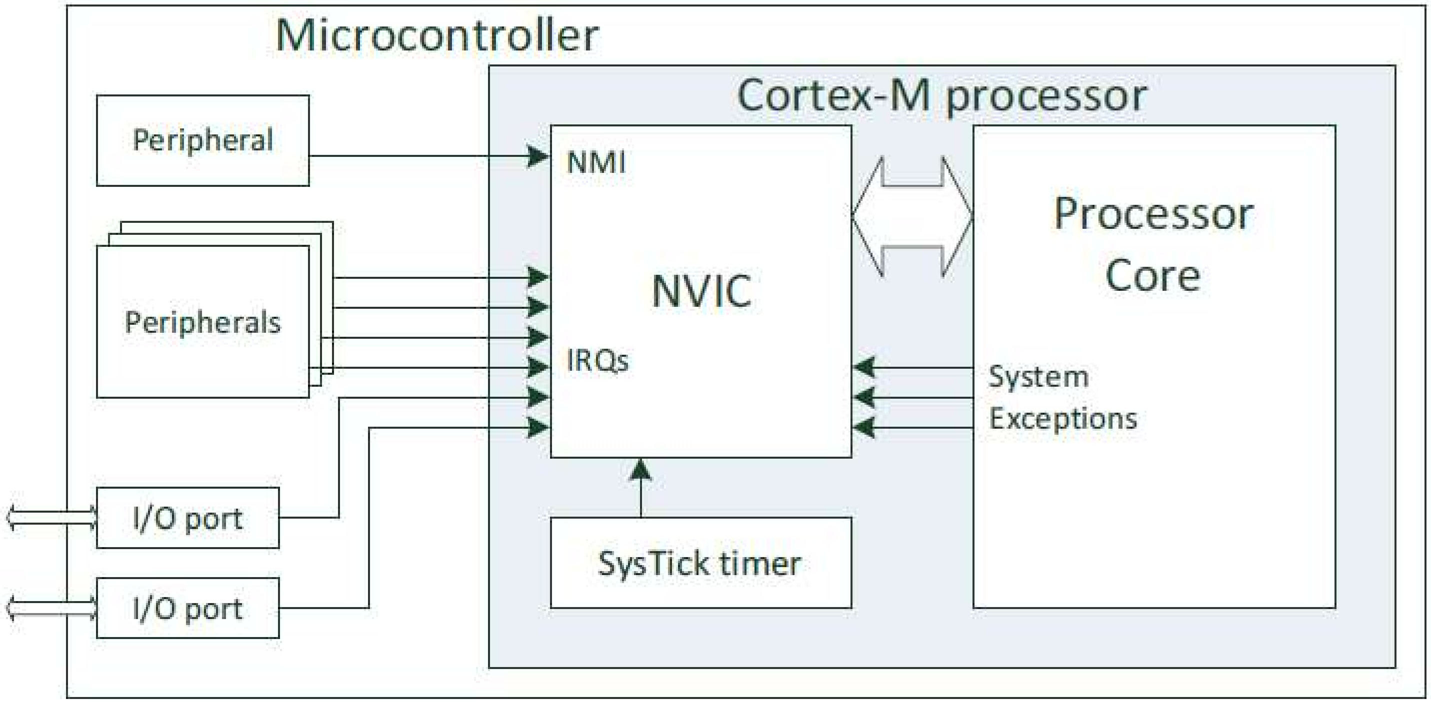
It is common to divide interrupts into multiple levels of priority.

While running an interrupt handler of a low priority interrupt, a higher priority interrupt can be triggered and get serviced. This is commonly known as nested interrupt.

Priority level of an interrupt can be programmable or fixed.

Apart from priority settings, some most interrupts can also be disabled or enabled by software.

Nested Vectored Interrupt Controller (NVIC)



NVIC handles the management of interrupt requests. It decides on which IRQ will be processed or suspended in case of nested interrupts.

Priority of the Interrupts in the Cortex-M0 and Cortex-M0+ processors

Exception number	Exception type	Priority	Descriptions
1	Reset	-3 (Highest)	Reset
2	NMI	-2	Non-Maskable Interrupt
3	HardFault	-1	Fault handling exception
4—10	Reserved	NA	—
11	SVCall	Programmable	Supervisor call via SVC instruction
12—13	Reserved	NA	—
14	PendSV	Programmable	Pendable request for system service
15	SysTick	Programmable	System Tick Timer
16	Interrupt #0	Programmable	External Interrupt #0
17	Interrupt #1	Programmable	External Interrupt #1
...	...	...	...
47	Interrupt #31	Programmable	External Interrupt #31

- The one with the lowest number has the highest priority
- The priority levels of some interrupts are fixed and some are programmable.
- NVIC in The Cortex-M0 and Cortex-M0+ processors supports up to 32 external IRQ inputs, an NMI input, and a number of system exceptions from within the processor.

Interrupt types

The Non-Maskable Interrupt (NMI) is similar to IRQ, but it cannot be disabled and has the highest priority apart from the reset.
It is very useful for safety critical systems like industrial control or automotive. Depending on the design of the microcontroller, the NMI could be used for power failure handling, or can be connected to a watchdog unit to restart a system if the system stopped responding.

HardFault is dedicated for handling fault conditions during program execution.
These fault conditions could be trying to execute an unknown opcodes, fault on bus interface or memory system, or illegal operations.

The SysTick Timer generates periodic interrupt for system maintenance

Interrupts #0...#31 the number of supported interrupts in Cortex-M0 or Cortex-M0+ processors could be from 1 to 32. The interrupt signals could be connected from on-chip peripherals (e. g., Timers, USART, I2C, SPI), or from external source via the I/O port.

SVCall (Supervisor Call) and Pendable Service Call (PendSV) for applications with Operating System.

Vector Table

After receiving an IRQ, the processor will need to decide whether to accept the request, and if yes, it will need to execute the corresponding interrupt handler.

And to do that, it will need to know the starting address of the handler, and the vector table is a lookup table in the memory that provides such information.

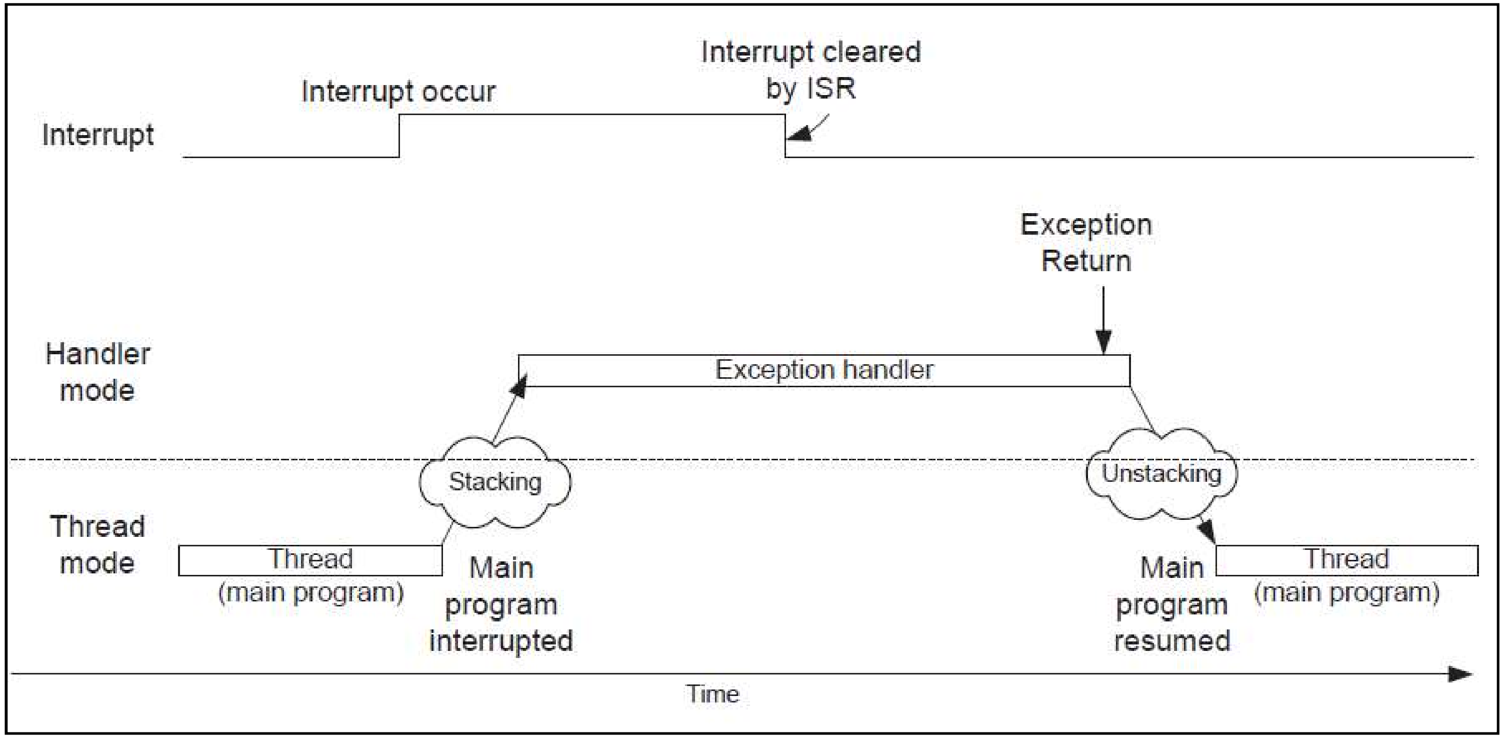
In the Cortex-M processors, the vector table stores the starting address of each interrupt individually

The built-in interrupt controller (NVIC) decides which interrupt to be serviced first based on the priority levels and generate a vector so that the processor can look up the starting address of the interrupt handler from the vector table

Memory Address		Exception Number
0x0000004C	Interrupt#3 vector	19
0x00000048	Interrupt#2 vector	18
0x00000044	Interrupt#1 vector	17
0x00000040	Interrupt#0 vector	16
0x0000003C	SysTick vector	15
0x00000038	PendSV vector	14
0x00000034	Not used	13
0x00000030	Not used	12
0x0000002C	SVC vector	11
0x00000028	Not used	10
0x00000024	Not used	9
0x00000020	Not used	8
0x0000001C	Not used	7
0x00000018	Not used	6
0x00000014	Not used	5
0x00000010	Not used	4
0x0000000C	HardFault vector	3
0x00000008	NMI vector	2
0x00000004	Reset vector	1
0x00000000	MSP initial value	0

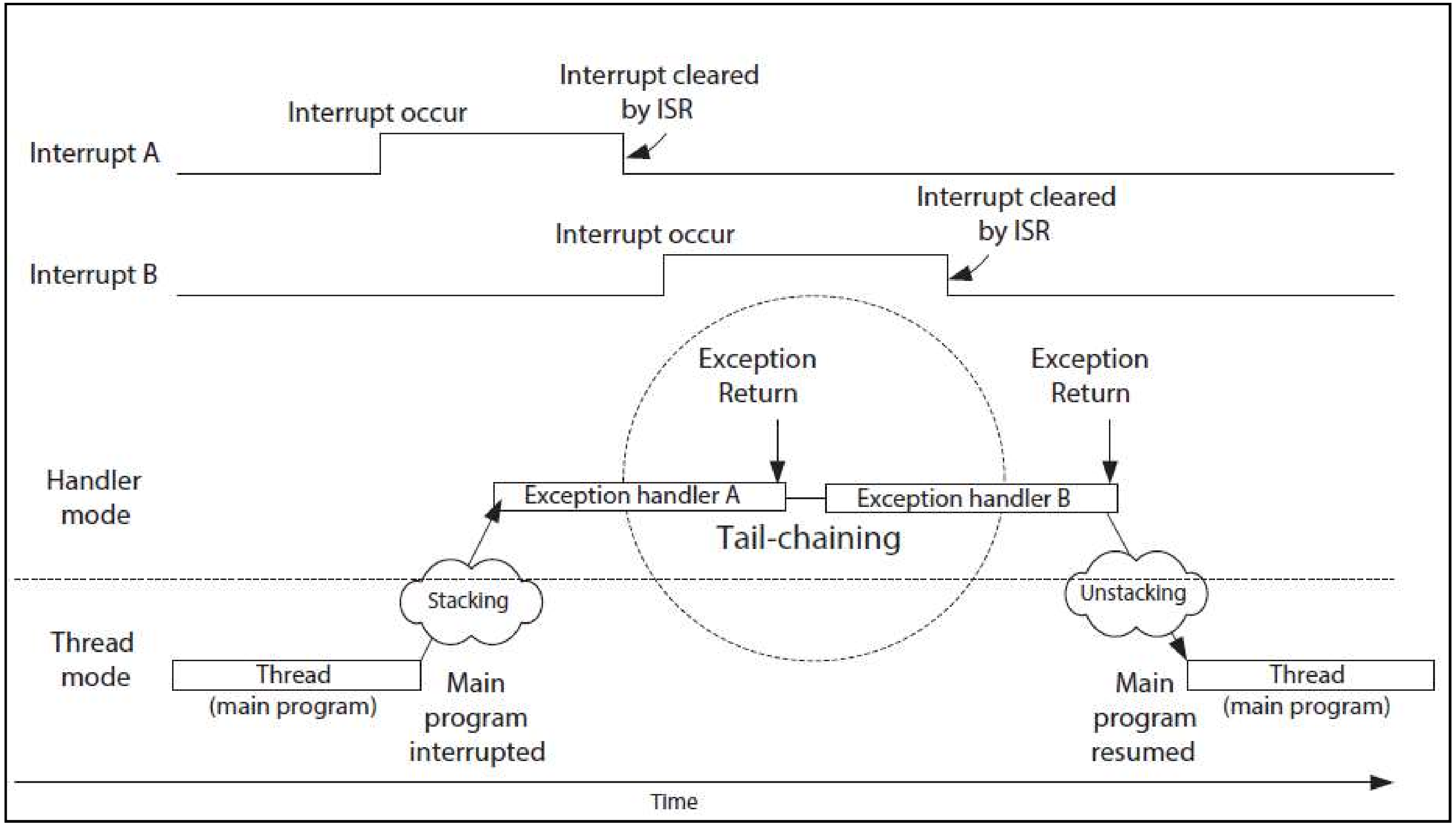
Figure 8.5
Vector table.

Stacking and unstacking of registers at interrupt entry and exit



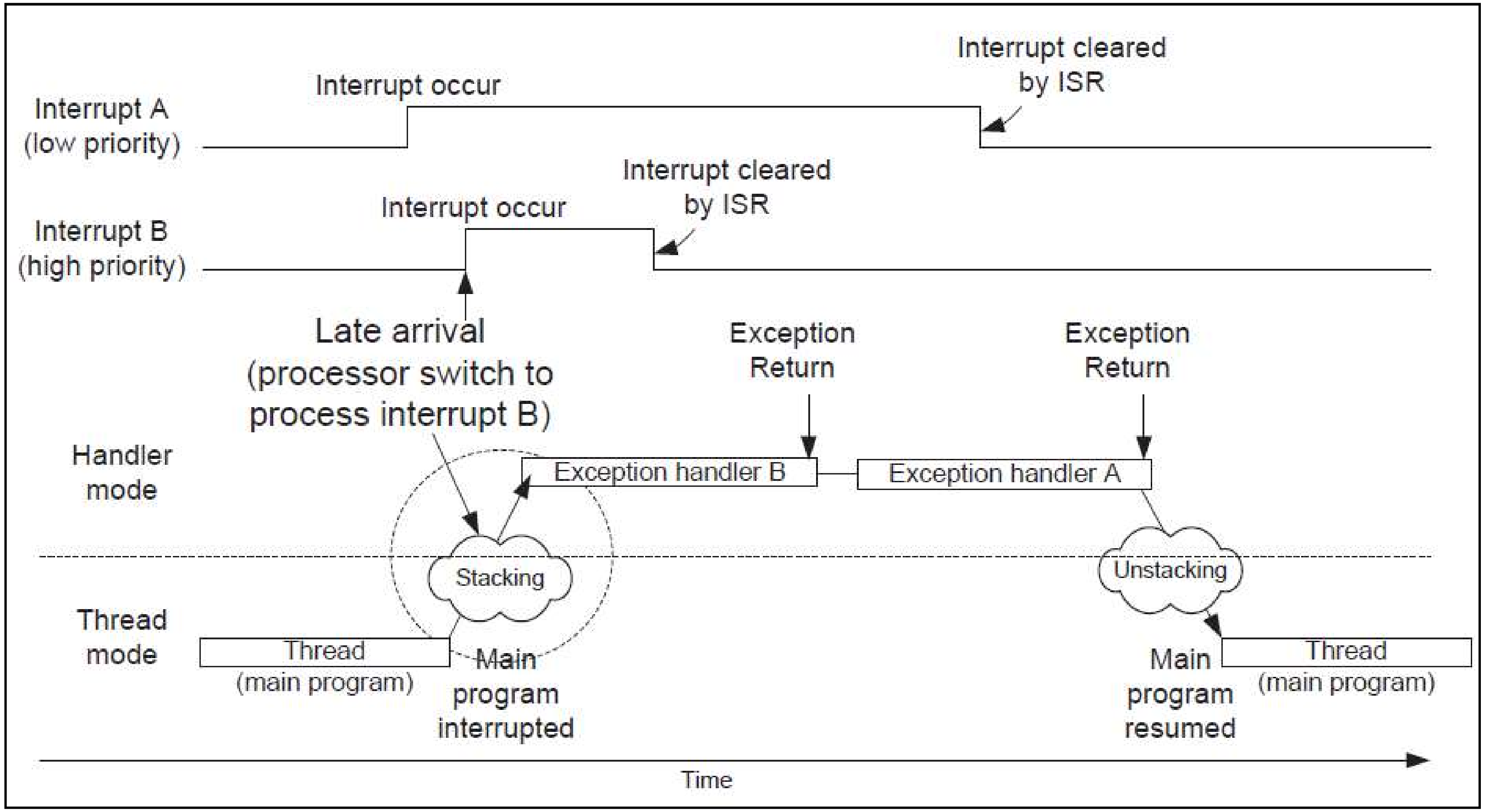
The actions of automatic saving and restoring of the register contents are called “stacking” and “unstacking”. This is required to allow an interrupted program to be resumed correctly.

Tail chaining



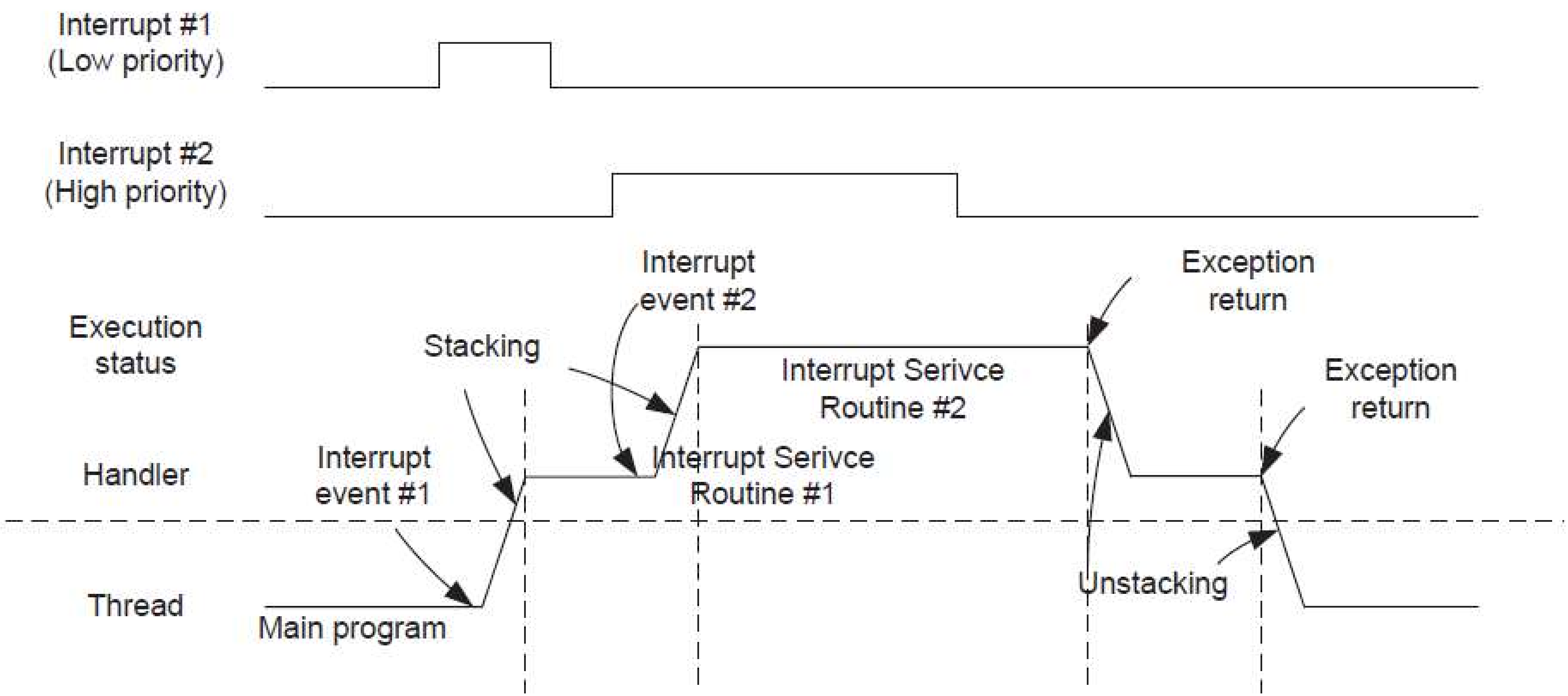
If an interrupt is in pending state when another interrupt handler is completed, instead of returning to the interrupted program and then entering interrupt sequence again, a tailchain scenario will occur. (Assume Interrupt A has a higher priority)

Late arrival

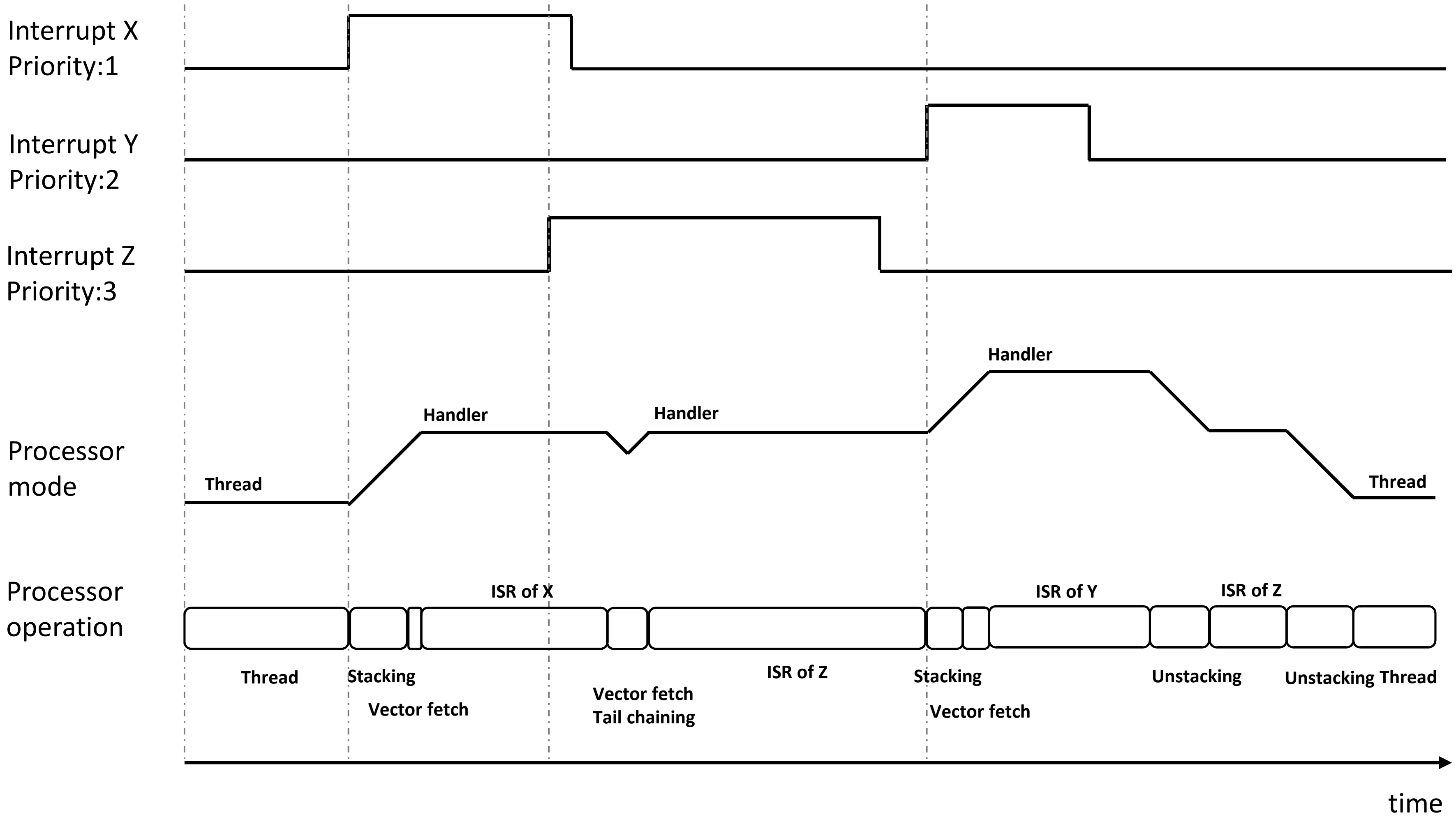


If a higher priority exception occurs during stacking process of a lower-priority exception, the processor switches to handle the higher priority exception first

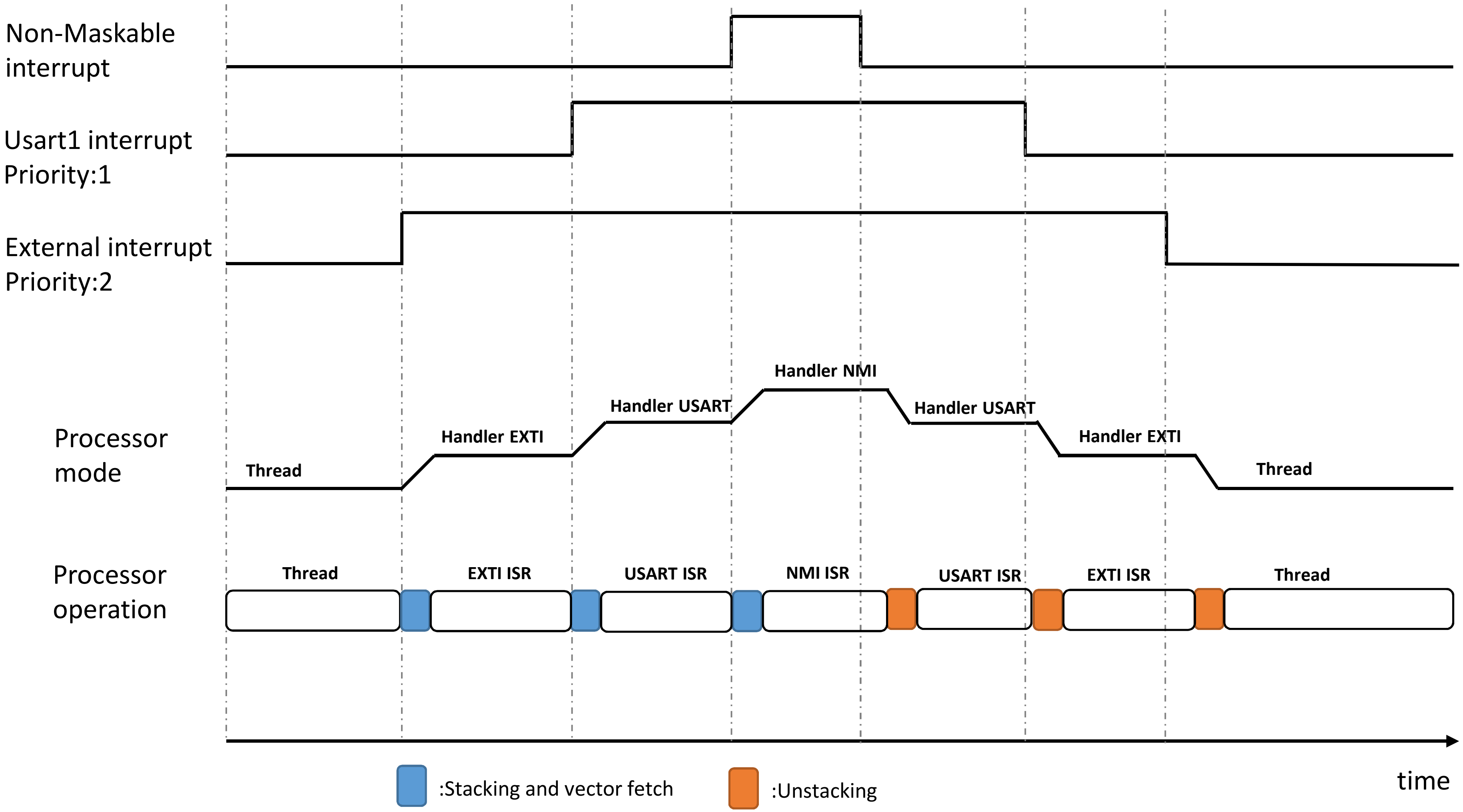
Nested interrupts



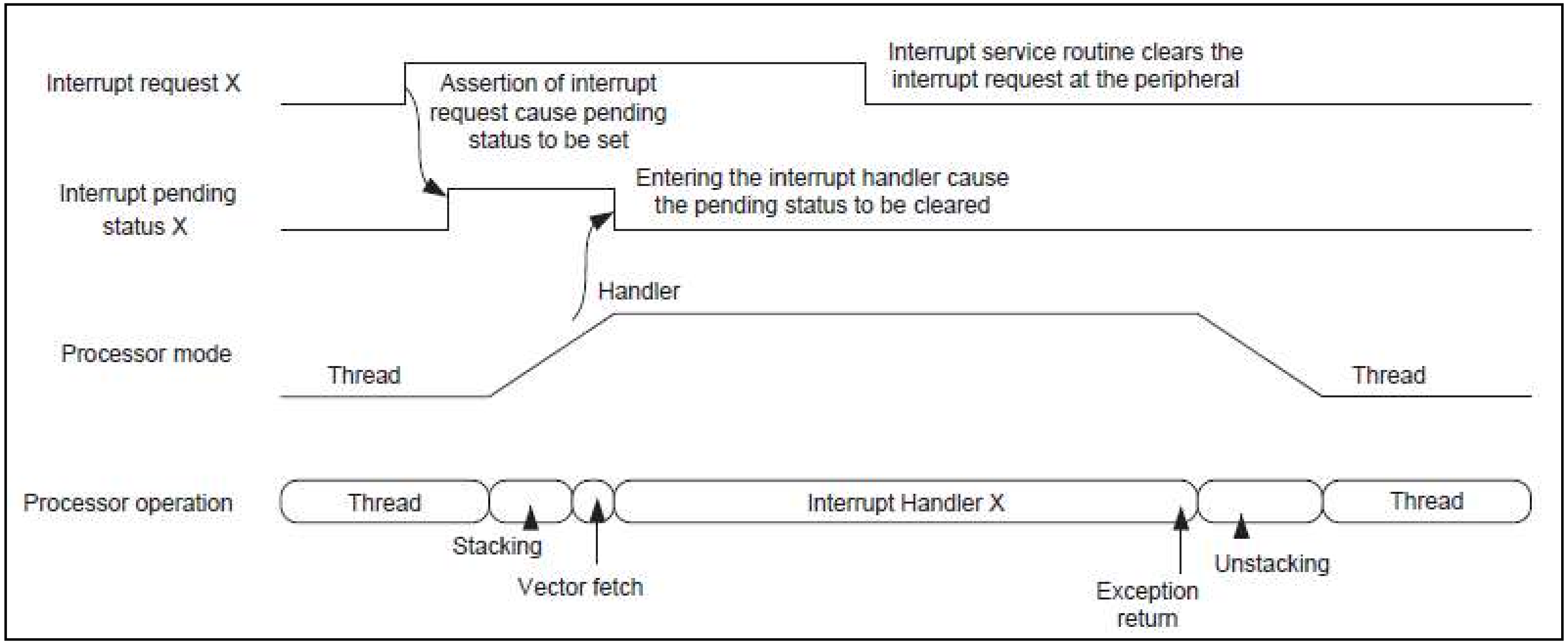
More nested interrupts



More nested interrupts

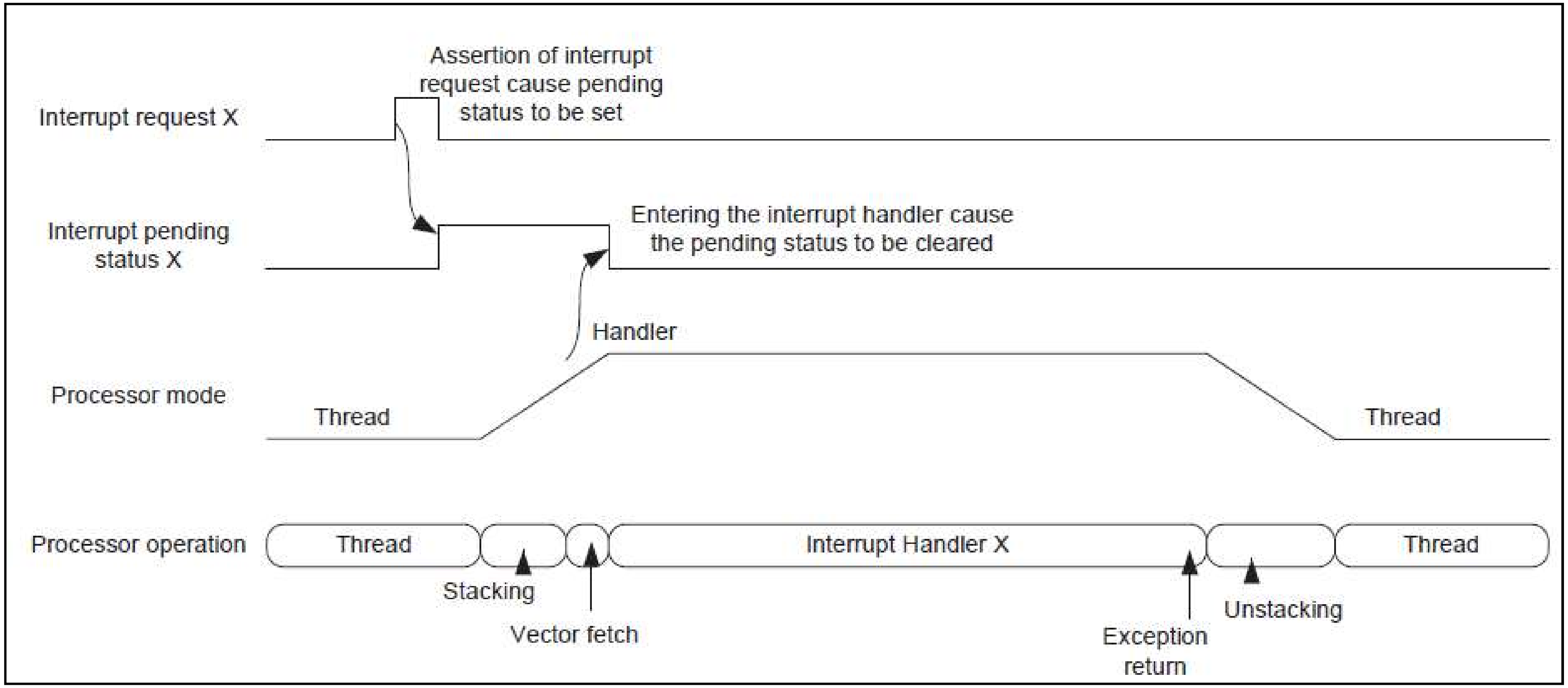


Interrupt Inputs and Pending Behavior



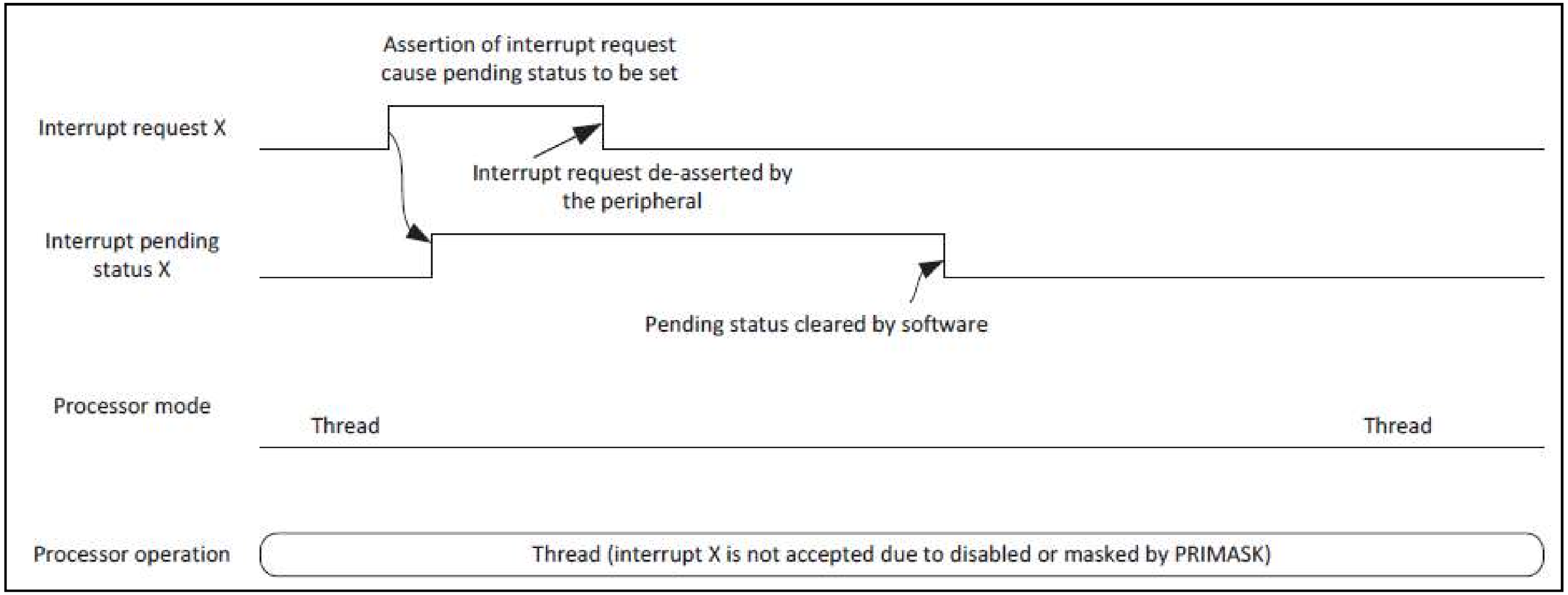
When the exception starts being served by the processor, the pending status is cleared automatically by hardware.

Simple Pulse Interrupt Handling



Some interrupt sources might generate IRQs in form of a pulse (for at least one clock cycle). In this case, the pending status register will hold the request until the interrupt is being served.

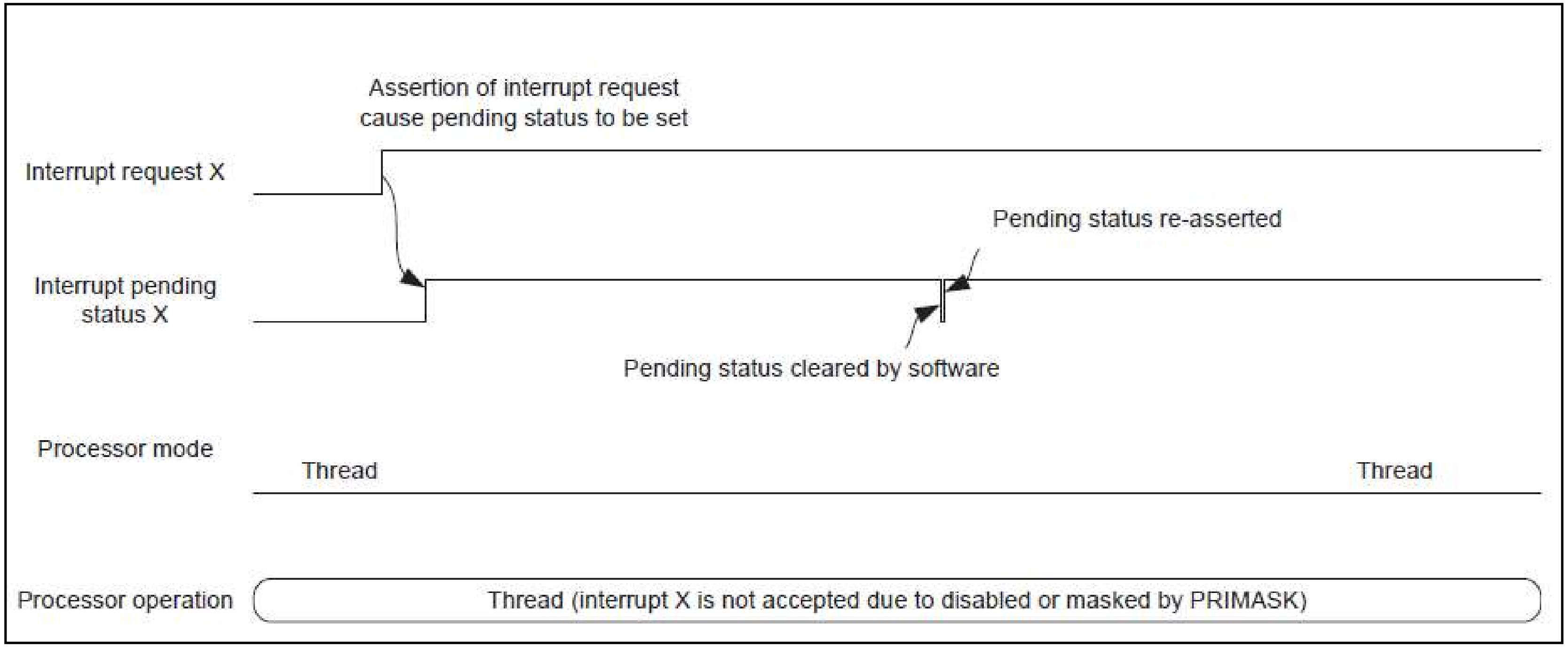
Canceling of Interrupt Pending Status Before the Interrupt Is Serviced



If the IRQ is not carried out immediately and is de-asserted, and the pending status is cleared by software, then the IRQ will be ignored, and the processor will not execute the interrupt handler

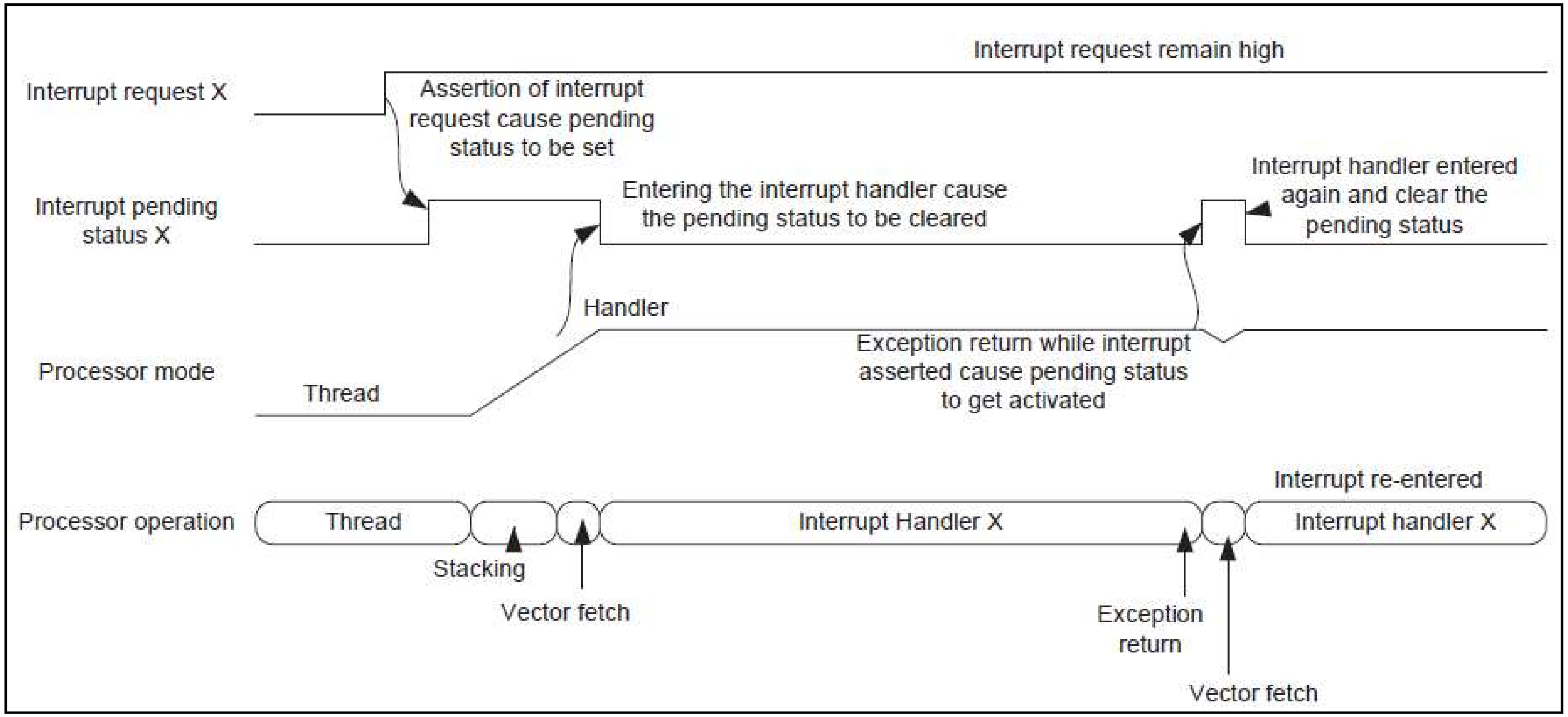
PRIMASK can be used to mask all interrupts. The PRIMASK is a single-bit register. When set to 1, only NMI and HardFault exceptions are allowed.

Clearing of Pending Status While Peripheral Still Asserting IRQ



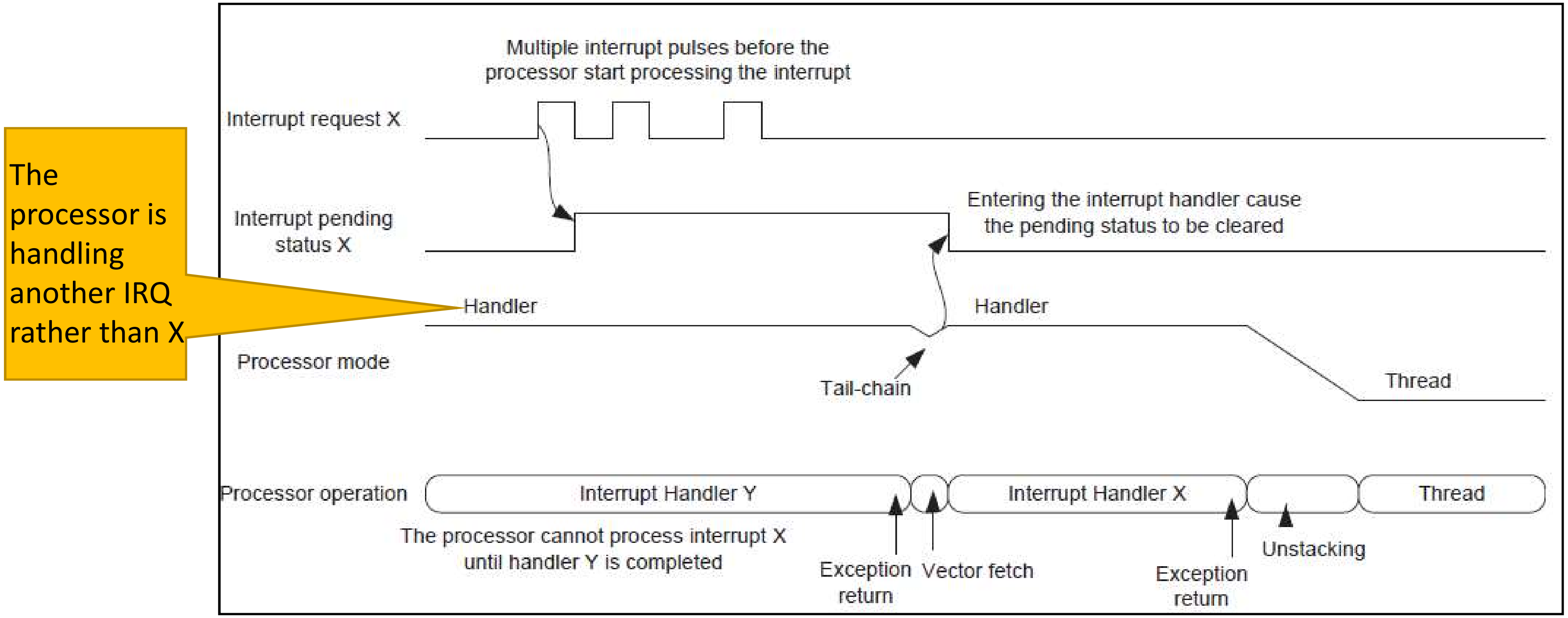
If the IRQ signal is still asserted by the peripheral when the software clears the pending status, the pending status will be asserted again immediately

IRQ Remains High When ISR Completed



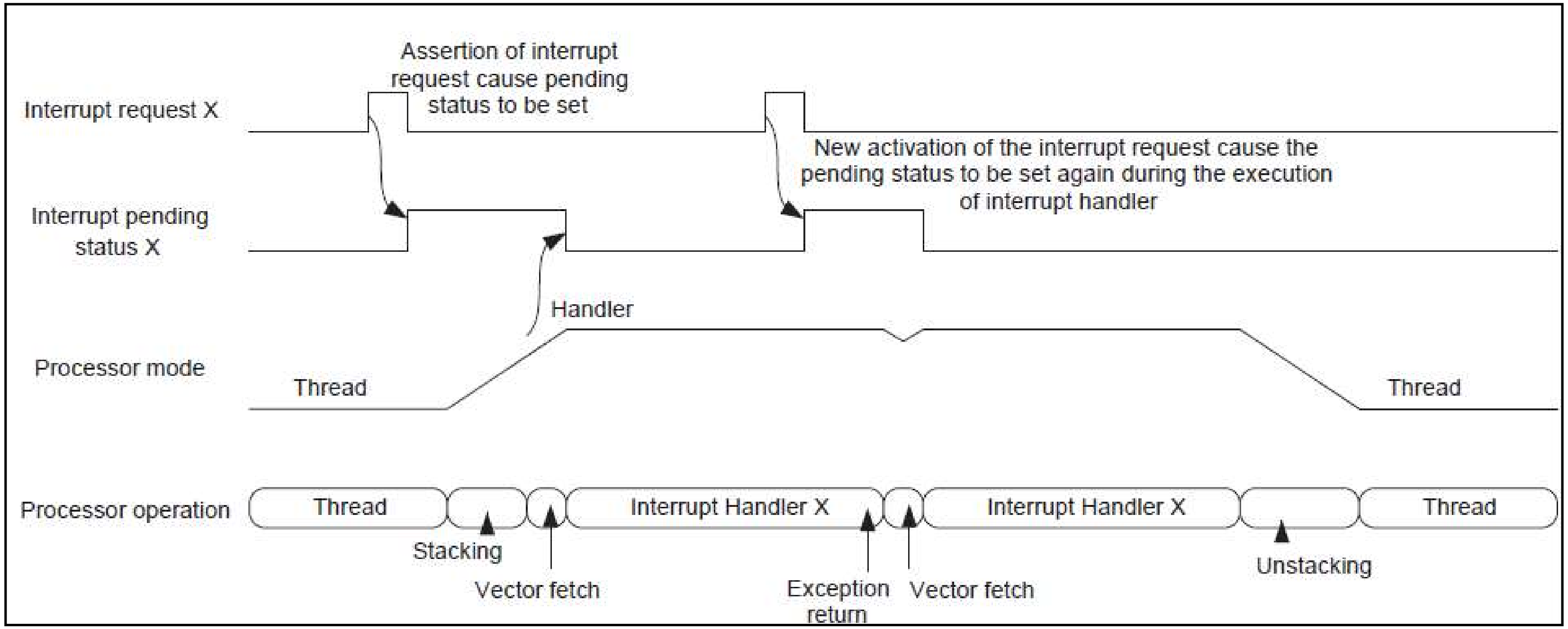
If the IRQ from a peripheral is not cleared during the execution of the exception handler, the pending status will be activated again at the exception return and will cause the exception handler to be executed again.

Multiple IRQ Pulses Before Entering ISR



For pulsed interrupts, if the IRQ is pulsed several times before the processor starts the ISR (for example, the processor could be handling another IRQ), then the multiple interrupt pulses will be treated as just one IRQ

IRQ Pulse During ISR Execution



If the pulsed IRQ is triggered again during the execution of the ISR, it will be processed as a new IRQ and will cause the ISR to be entered again after the interrupt exit.

NVIC in STM32F4

- 82 maskable interrupt channels including external interrupt lines

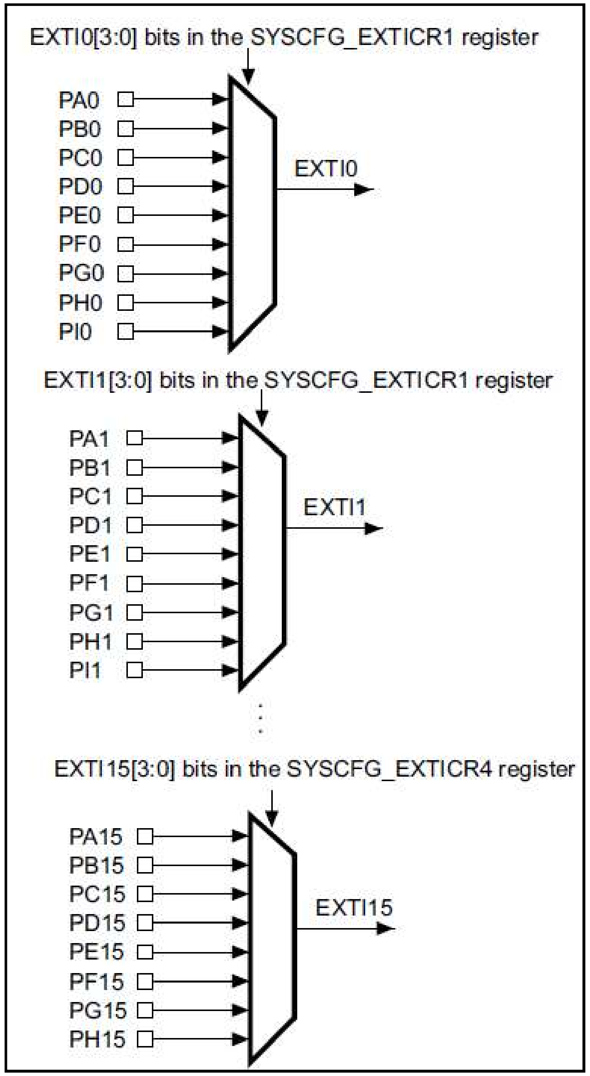
- Check vector table from STM32F4 reference manual

Table 62. Vector table for STM32F42xxx and STM32F43xxx (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	-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	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	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_Stream0	DMA1 Stream0 global interrupt	0x0000 006C
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074

.
.
.

External interrupt (EXTI) lines in STM32F4



There are 16 external interrupt/event lines in this manner.

Only one pin can be a source of interrupt for each external interrupt.

For example, we cannot define both PA0 and PB0 as input interrupt pins.

The seven other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB OTG FS Wakeup event
- EXTI line 19 is connected to the Ethernet Wakeup event
- EXTI line 20 is connected to the USB OTG HS Wakeup event
- EXTI line 21 is connected to the RTC Tamper and TimeStamp
- EXTI line 22 is connected to the RTC Wakeup event

EXTI registers

12.3.1 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									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **MRx:** Interrupt mask on line x

0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

EXTI registers

12.3.3 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									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **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

EXTI registers

12.3.4 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									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **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.

EXTI registers

12.3.5 Software interrupt event register (EXTI_SWIER)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
									rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **SWIERx:** Software Interrupt on line x

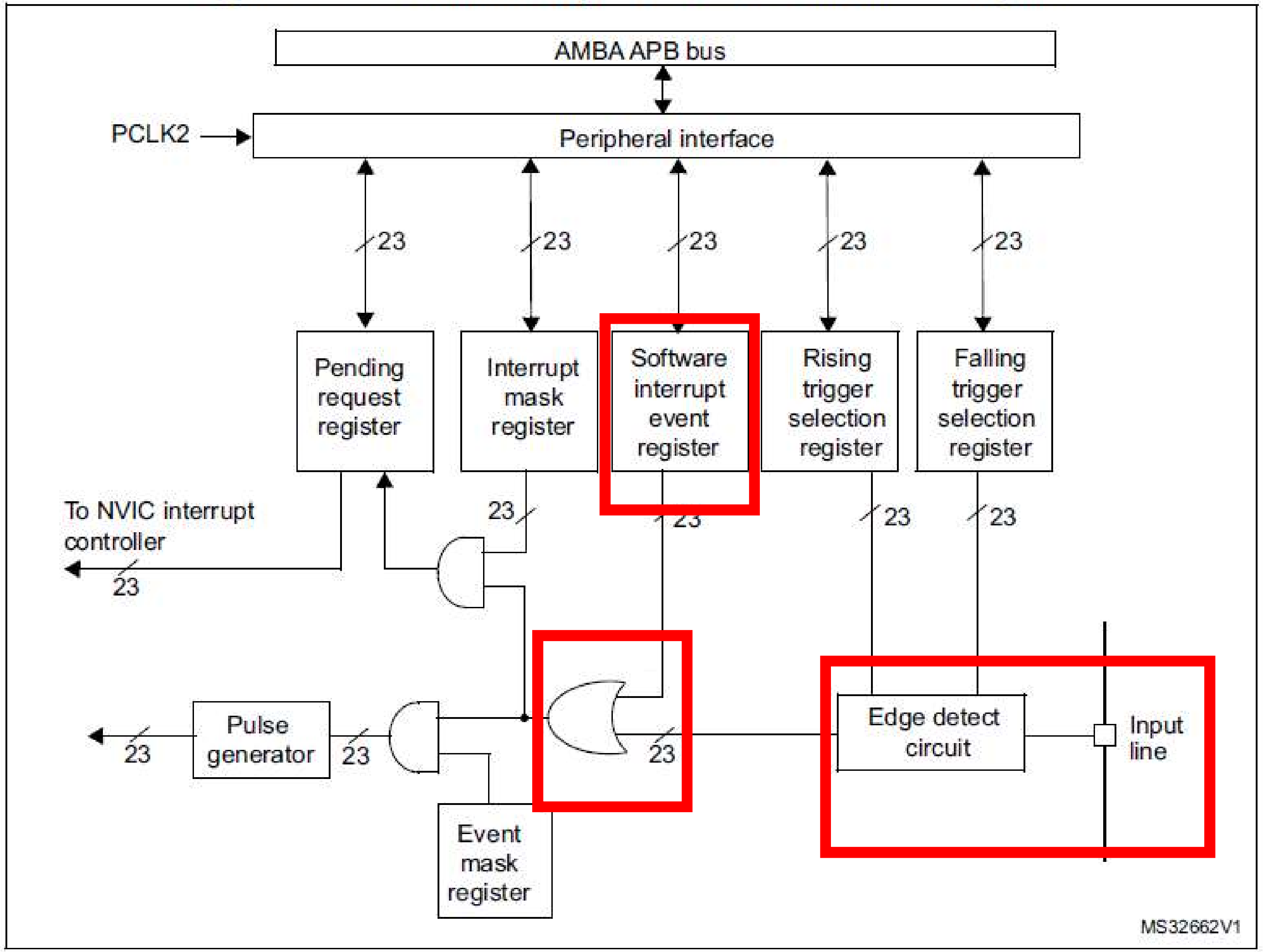
If interrupt are enabled on line x in the EXTI_IMR register, writing '1' to SWIERx bit when it is set at '0' sets the corresponding pending bit in the EXTI_PR register, thus resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to the bit).

EXTI registers

Software interrupt event register and External Input line are connected to the **OR** gate.

Figure 41. External interrupt/event controller block diagram



EXTI registers

12.3.6 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									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **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 programming it to '1'.

read/clear (rc_w1) Software can read as well as clear this bit by writing 1. Writing '0' has no effect on the bit value.