

Exceptions & Interrupts

Microcontroller Application and Development

Sorayut Glomglome

Outline

1. Exception
2. Interrupt
3. Mechanism
4. Priority
5. Interrupt Service Routine
6. STM32 Interrupt Module

Learning Outcomes

1. Understanding Interrupt Mechanism
2. Understanding EXTI

Polling vs Interrupt



polling



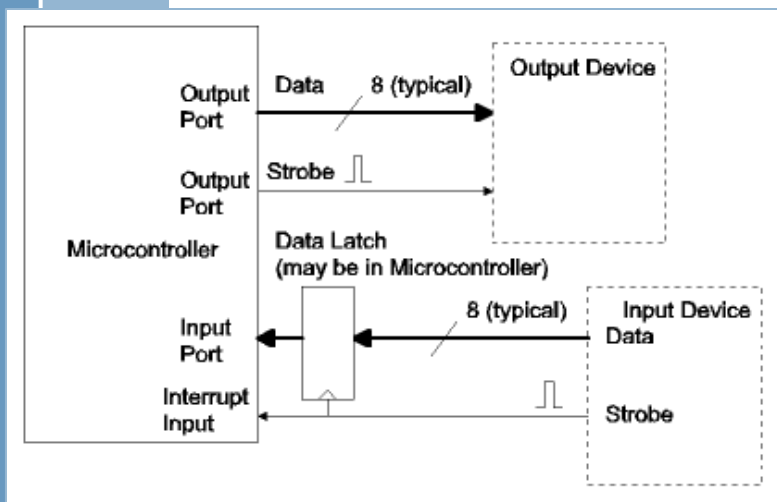
interrupt

What is an exception?

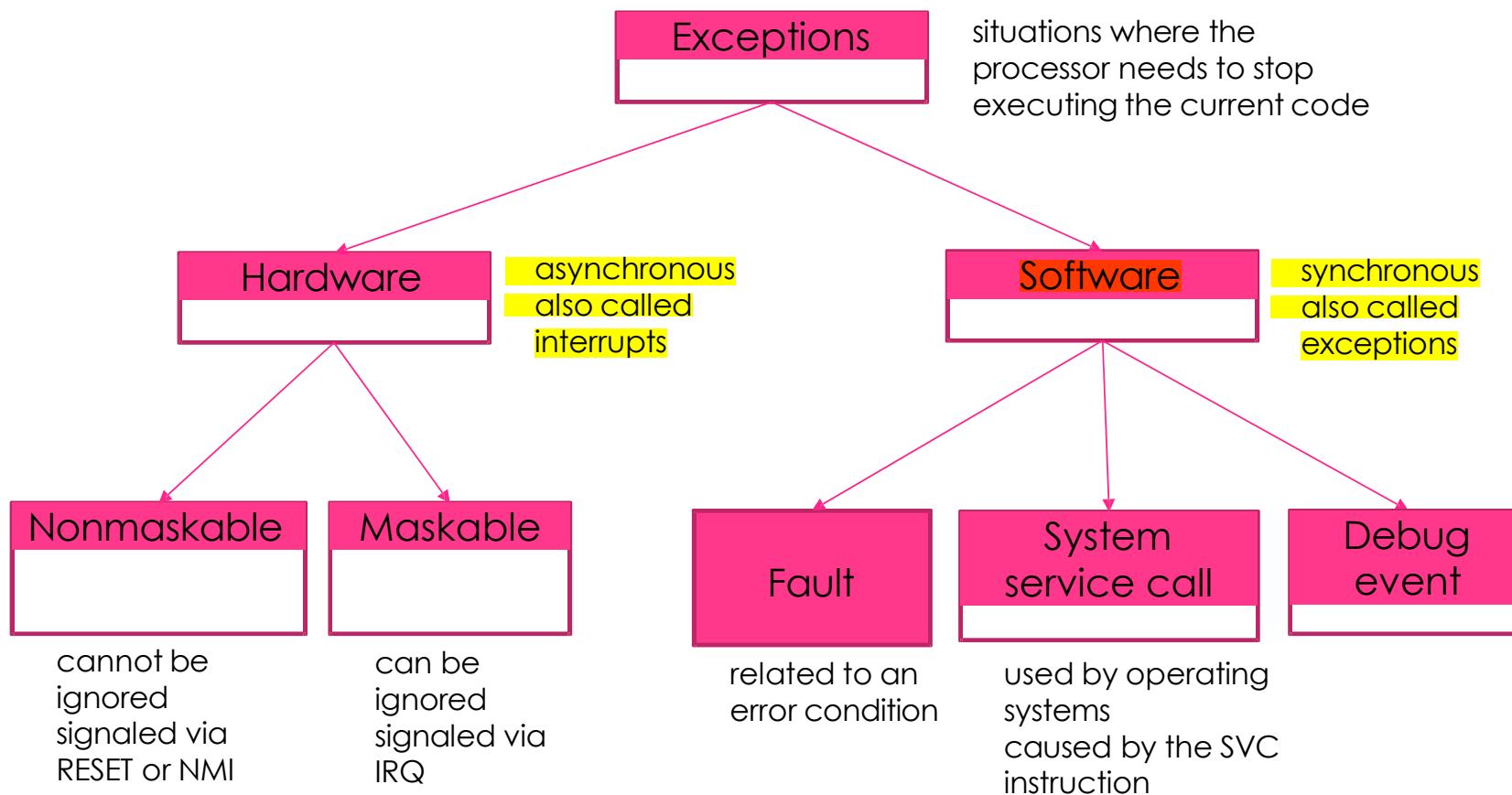
- A special event that requires the CPU to stop normal program execution and perform some service related to the event.
- Examples of exceptions include
 - I/O completion, timer time-out, end of conversion,
 - illegal opcodes, arithmetic overflow, divide-by-0, etc.

Functions of exceptions

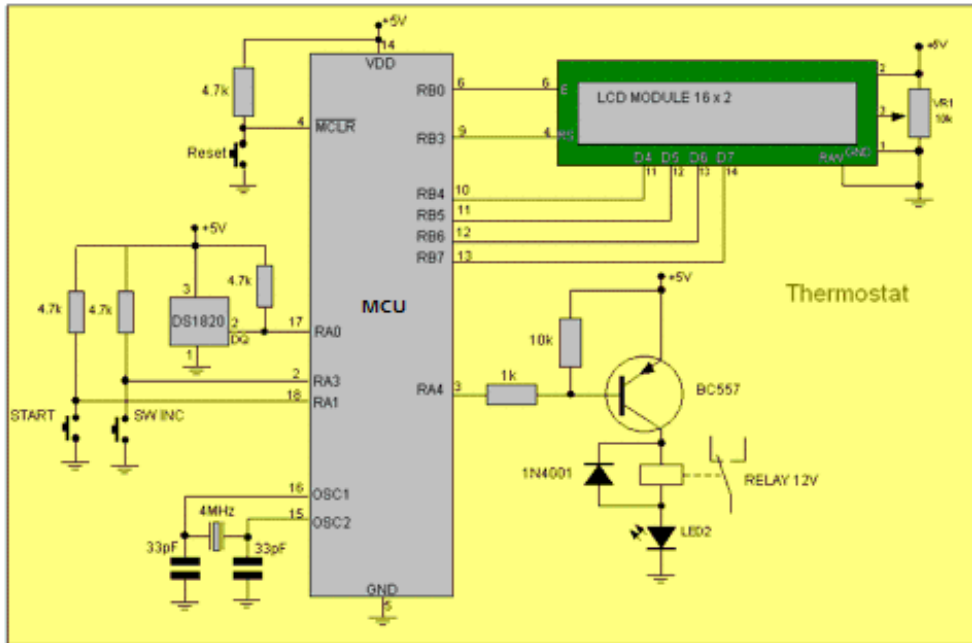
- Respond to infrequent but important events
 - Alarm conditions like low battery power
 - Error conditions
- I/O synchronization
 - Trigger interrupt when signal on a port changes
- Periodic interrupts
 - Generated by the timer at a regular rate
 - SysTick timer can generate interrupt when it hits zero
 - Reload value + frequency determine interrupt rate



Types of Exceptions




Interrupt



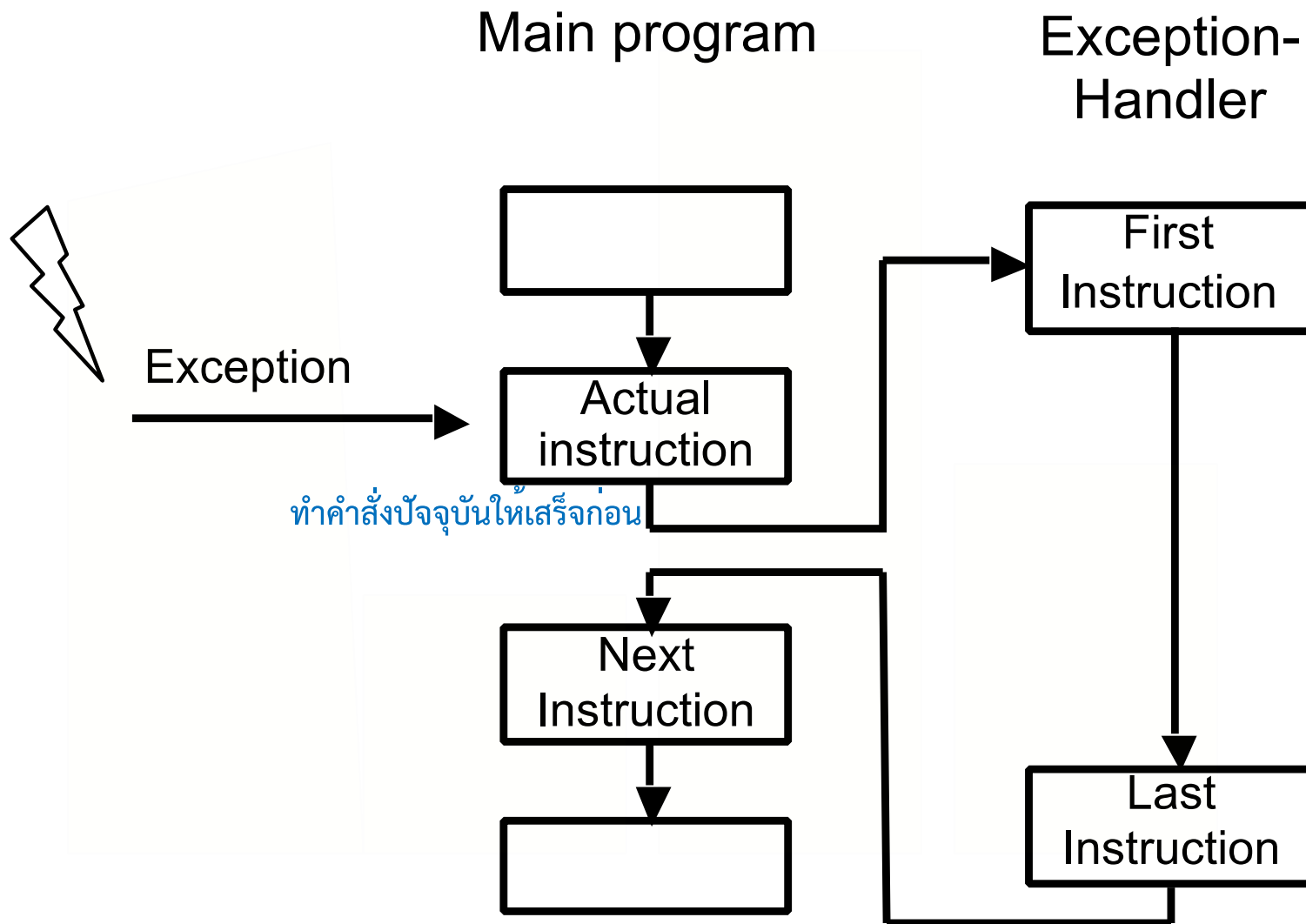
Programmable Room
Temperature Control

- Interrupt is a powerful concept in embedded systems for separating the time-critical events from the others and execute them in a prioritized manner.
- In a typical embedded system, the embedded processor (microcontroller) is responsible for doing more than one task (but can do only one at a time).

Interrupt Service Cycle (Overhead)

1. Saving the program counter value in the stack
2. Saving the CPU status (including the CPU status register and some other registers) in the stack
3. Identifying the cause of interrupt
4. Resolving the starting address of the corresponding interrupt service routine
5. Executing Interrupt Service Routine 
6. Restoring the CPU status and the program counter from the stack
7. Restarting the interrupted program

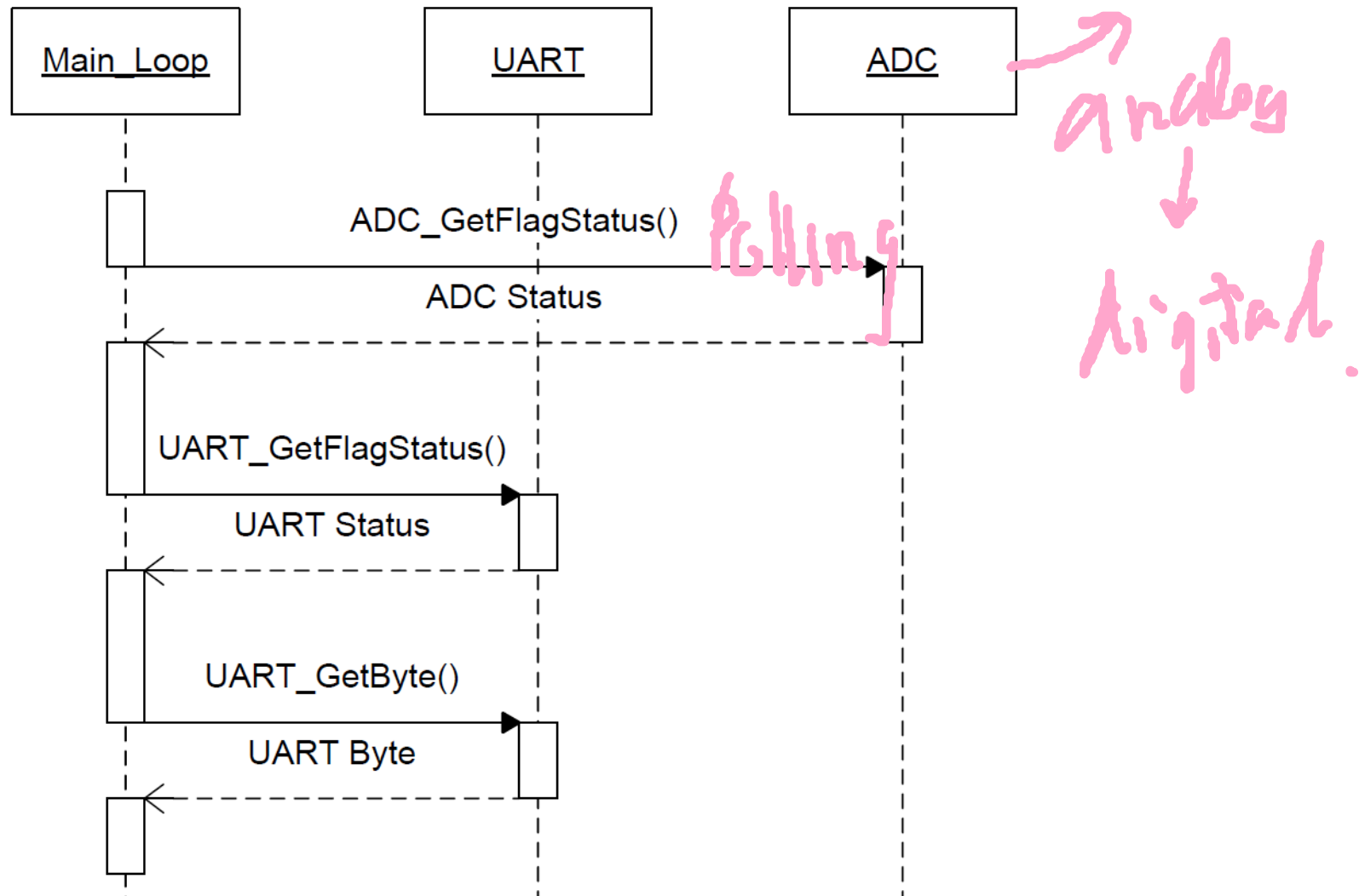
Program execution when an exception occurs



Difference between the ISR and the standard function calls

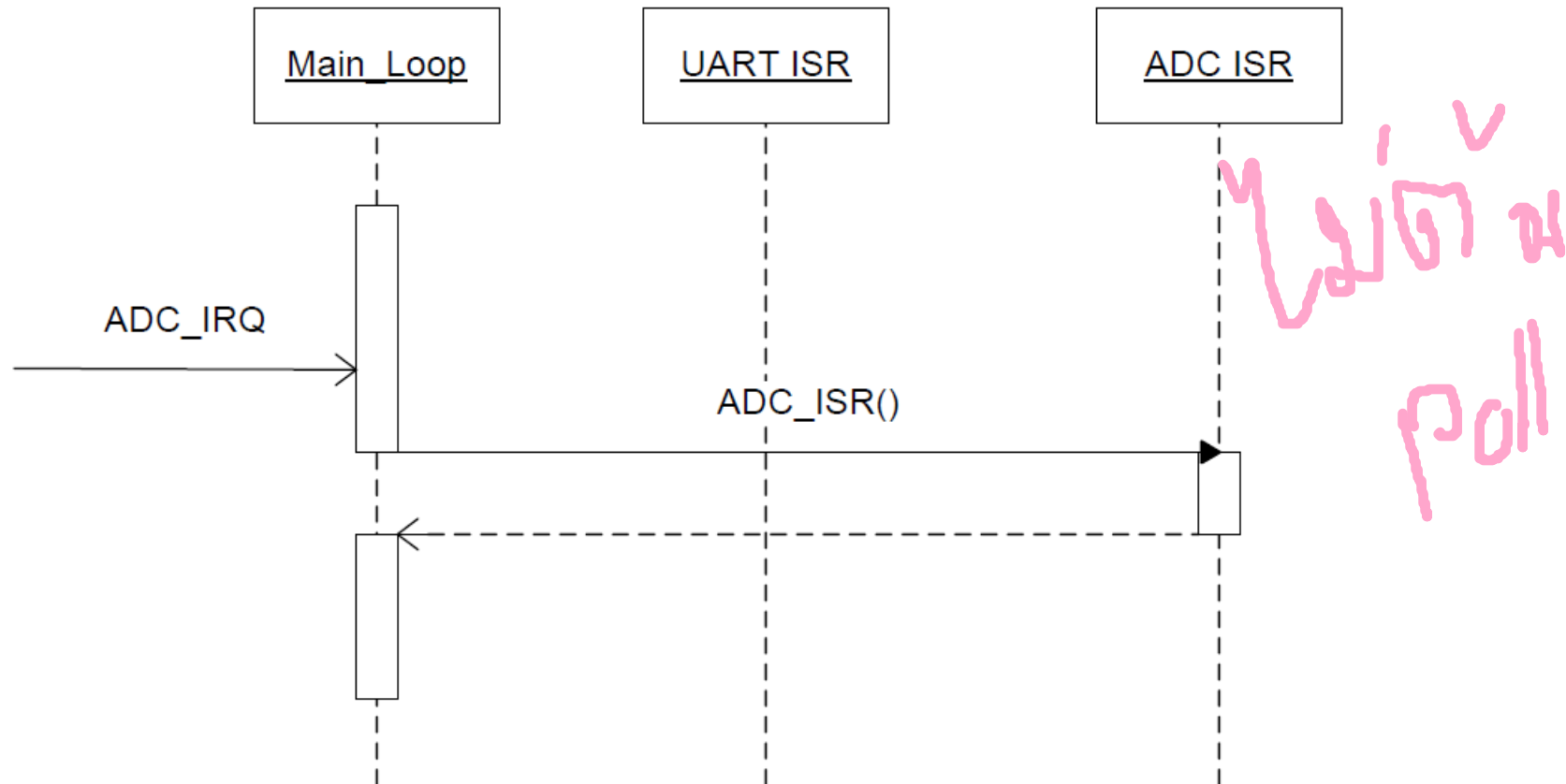
- The standard function calls are realized in a synchronous manner with branch instructions
- Interrupt service routines are called when an exception signals occur
 - Vector table contains the addresses of the interrupt service routines

Programming without interrupts



➔ The `main()` function executes all peripheral calls in a fixed sequence

Programming with interrupts



Interrupt and exception vectors

- When an exception or an interrupt occurs
 - CPU interrupts the execution of the main program
 - CPU jumps to the vector address, which depends on the exception type
- Base address of the vector table is usually 0
- Cortex M3 contains both an interrupt and an exception vector tables

Exception states

Each exception is in one of the following states:

Inactive

The exception is not active and not pending

Pending

The exception is waiting to be serviced by the processor.

An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

Active

An exception that is being serviced by the processor but has not completed.

Note: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.

Active and pending

The exception is being serviced by the processor and there is a pending exception from the same source.

Cortex-M3 Exception Types



No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other handler not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
.....	settable
256	Interrupt#240	247	settable	External Interrupt #240

Vector Table

Exception number	IRQ number	Offset	Vector
255	239	0x03FC	IRQ239
.		.	.
.		.	.
.		.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13		0x0038	Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

Exception vector table of the Cortex M3

(Ref. RM0008 Reference manual)

Position	priority	Type of Priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-3	fixed	Reset	Reset	Reset	0x0000_0004
-2	fixed	UMI	Non maskable interrupt	Non maskable interrupt	0x0000_0008
-1	settable	HardFault	All class of fault	All class of fault	0x0000_000C
0	settable	MemManage	Memory management	Memory management	0x0000_0010
1	settable	BusFault	Pre-fetch fault, memory access fault	Pre-fetch fault, memory access fault	0x0000_0014
2	settable	UsageFault	Undefined instruction or illegal state	Undefined instruction or illegal state	0x0000_0018
-	-	-	Reserved	Reserved	0x0000_001C 0x0000_002B
3	settable	SCCall	System service call via SWI instruction	System service call via SWI instruction	0x0000_002C
4	settable	Debug Monitor	Debug Monitor	Debug Monitor	0x0000_0030
-	-	-	Reserved	Reserved	0x0000_0034
5	settable	PendSV	Pendable request for system service	Pendable request for system service	0x0000_0038
6	Settable	SysTick	System tick timer	System tick timer	0x0000_003C

Interrupt vector table of the Cortex M3 (1)

Position	priority	Type of Priority	Acronym	Description	Address
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
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

Interrupt vector table of the Cortex M3 (2)

Position	priority	Type of Priority	Acronym	Description	Address
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	CAN1_TX	CAN1 TX interrupts	0x0000_008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000_0094
22	29	settable	CAN1_SCE	CAN1 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
26	33	settable	TIM1_TRG_COM	TIM1 Trigger & Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC TIM1	Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0

Interrupt vector table of the Cortex M3 (3)

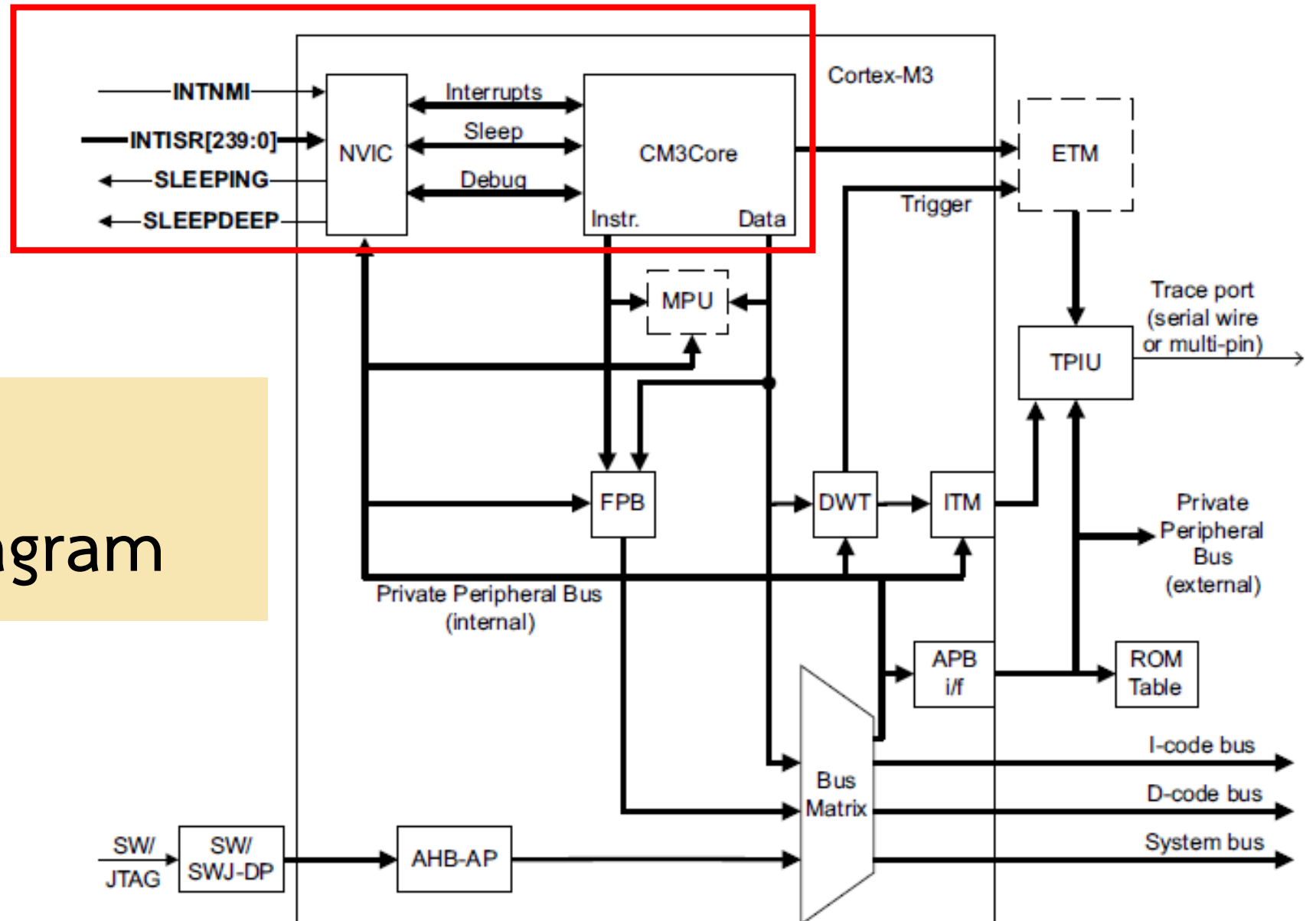
Position	priority	Type of Priorty	Acronym	Description	Address
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8
-	-	-	-	Reserved	0x0000_00EC 0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114

Pin 10
— Pin 15

Interrupt vector table of the Cortex M3 (4)

Position	priority	Type of Priority	Acronym	Description	Address
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4	DMA2 Channel4 global interrupt	0x0000_012C
60	67	settable	DMA2_Channel5	DMA2 Channel5 global interrupt	0x0000_0130
61	68	settable	ETH	Ethernet global interrupt	0x0000_0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000_0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000_013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000_0140
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000_0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000_0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000_014C

NVIC Block Diagram

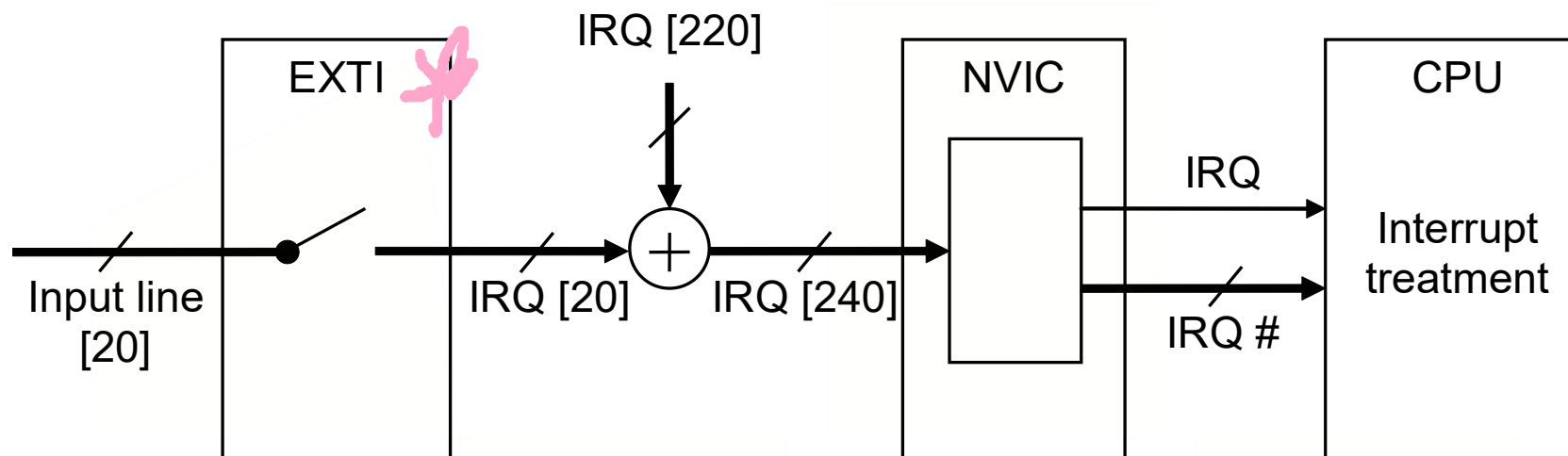


Nested Vectored Interrupt Controller (NVIC) *

- Features
 - 68 (not including the sixteen Cortex™-M3 interrupt lines) exceptions
 - 16 programmable priority levels (4 bits of interrupt priority are used)
 - Low-latency exception and interrupt handling
 - Power management control
 - Implementation of System Control Registers
- The NVIC and the processor core interface are closely coupled, which enables **low latency interrupt processing** and efficient processing of late arriving interrupts
- All interrupts including the core exceptions are managed by the NVIC.

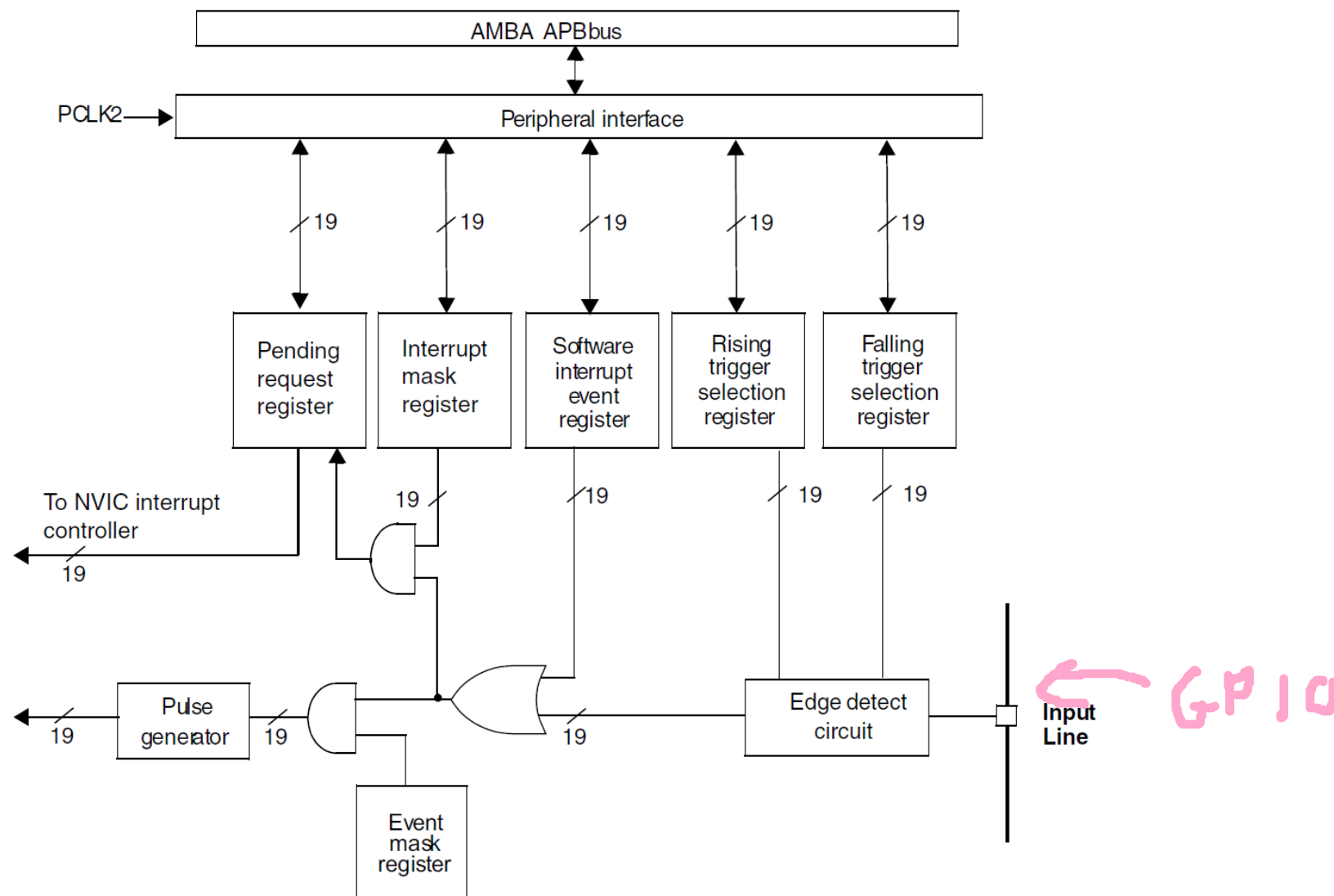
External interrupt/event controller (EXTI)

เกิดจาก GPIO

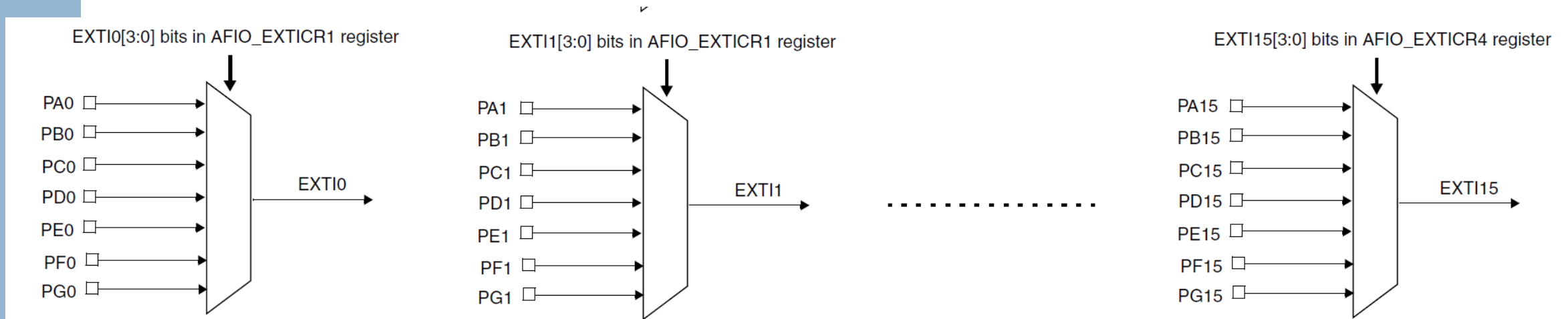


- EXTI consists of up to 20 edge detectors for generating event/interrupt requests
- Features
 - Independent trigger and mask on each interrupt/event line
 - Dedicated status bit for each interrupt line
 - Generation of up to 20 software event/interrupt request
 - Detection of external signal with pulse width lower than APB2 clock

EXTI Block diagram



External Interrupt/Event GPIO mapping



interrupt p0 จากทุก port

NVIC Registers

Each interrupt input has several registers to control it

- Enable/Disable Bit 240 bit
 - Enable or disable the interrupt, Can be set, cleared or read
- Pending Bit
 - If the pending bit is set, then the interrupt is pending
 - A pending interrupt can only be taken (become active) if it is enabled and it has sufficient priority to run
 - Pending bit can be set, cleared or read

NVIC Registers

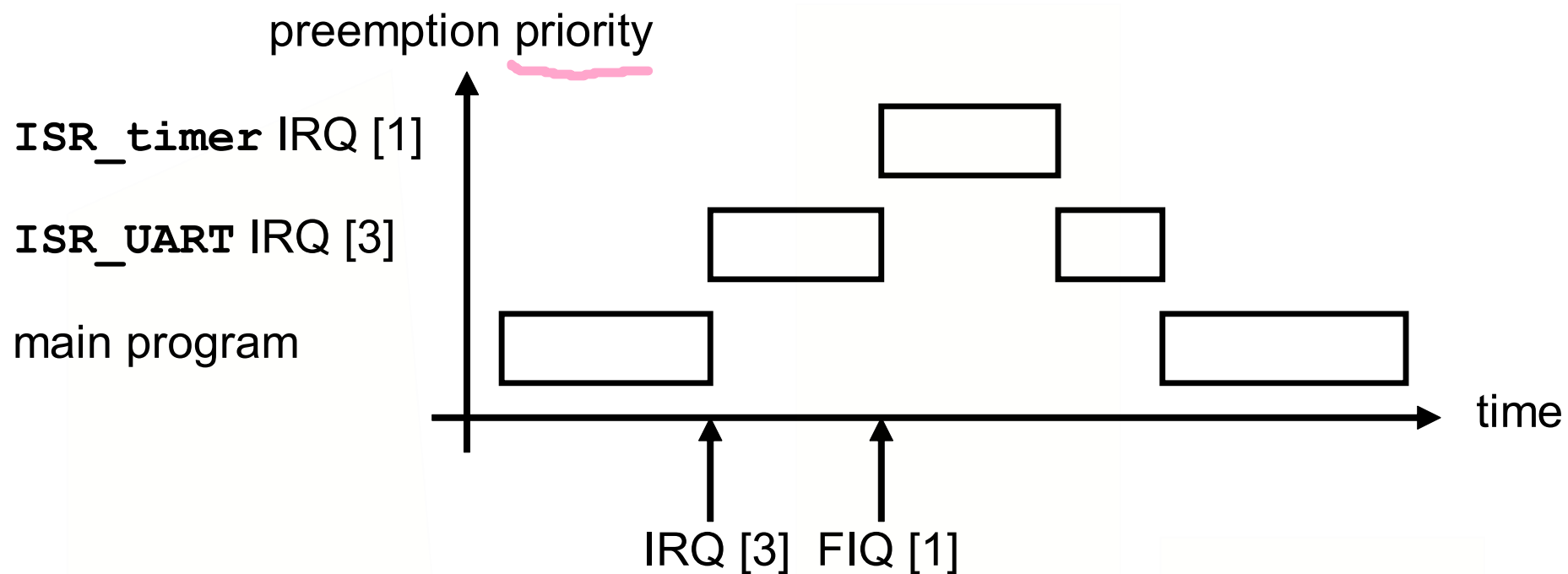
- Active Bit
 - A bit is set if the interrupt is executing or “active-stacked”
 - “Active-stacked” means the interrupt was executing, but was pre-empted by another higher-priority interrupt
 - Active register is normally read only
- Priority field *4 x 240 bits*
 - priority management for each interrupt

NVIC Register Descriptions

- › IRQ 0 to 239 Set-Enable Registers
 - Enable interrupts
 - Determine which interrupts are currently enabled
- › IRQ 0 to 239 Clear-Enable Registers
 - Disable interrupts
 - Determine which interrupts are currently disabled
- › IRQ 0 to 239 Set-Pending Register
 - Force interrupts into the pending state
 - Determine which interrupts are currently pending
- › IRQ 0 to 239 Clear-Pending Register
 - Clear pending interrupts
 - Determine which interrupts are currently pending

Preemption of Interrupt

การยัด



Priorities of the exceptions

- NVIC supports software-assigned priority levels
 - Priority level from 0 to 255 can be assigned to each hardware Interrupt
 - **PRI_N** field of the *Interrupt Priority Register*
- All priority levels can be split into a preemption and a sub priorities
 - **PRIGROUP** field of the *Application Interrupt and Reset Control Register*

PRIGROUP[2:0]	Binary point position	^① Pre-emption field	^② Subpriority field	Number of pre-emption priorities	Number of subpriorities
b000	bxxxxxxx.y	[7:1] 7bit	[0] 1bit	128 2 ⁷	2 2
b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
b011	bxxxx.yyyy	[7:4]	[3:0]	16	16
b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
b110	bx.yyyyyyy	[7]	[6:0]	2	128
b111	b.yyyyyyyy	None	[7:0]	0	256



NVIC Config

tion Clock Configuration Project Manager Tools

Additional Softwares Pinout

NVIC Mode and Configuration

Configuration

3

✓ NVIC ✓ Code generation

Priority Group 1 bits for pre-emption priority 3 bits for sub... ☐ Sort by Preemption Priority and Sub Priority

Search Search (Ctrl+F) ☐ Show only enabled interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
4 EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	1	0
FPU global interrupt	<input type="checkbox"/>	0	0

System Core A

DMA ⚠

GPIO ✓

2 NVIC ✓

RCC ✓

```
void HAL_MspInit(void)
{
    /* USER CODE BEGIN MspInit 0 */

    /* USER CODE END MspInit 0 */

    HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_1);

    /* System interrupt init*/
    /* SysTick_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);

    /* USER CODE BEGIN MspInit 1 */

    /* USER CODE END MspInit 1 */
}
```

EXTI Config

Pinout & Configuration | Clock Configuration | Project Manager

Additional Software | Pinout

GPIO Mode and Configuration

Configuration

☐ Group By Peripherals

☒ GPIO ☒ NVIC

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin	Signal	GPIO o...	GPIO m...	GPIO P...	Maximu...	Fast Mo...	User La...	Modified
PC13	n/a	n/a	Externa...	No pull...	n/a	n/a		<input type="checkbox"/>

PC13 Configuration :

GPIO mode

GPIO Pull-up/Pull-d...

User Label

```
void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();

    /*Configure GPIO pin : PC13 */
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

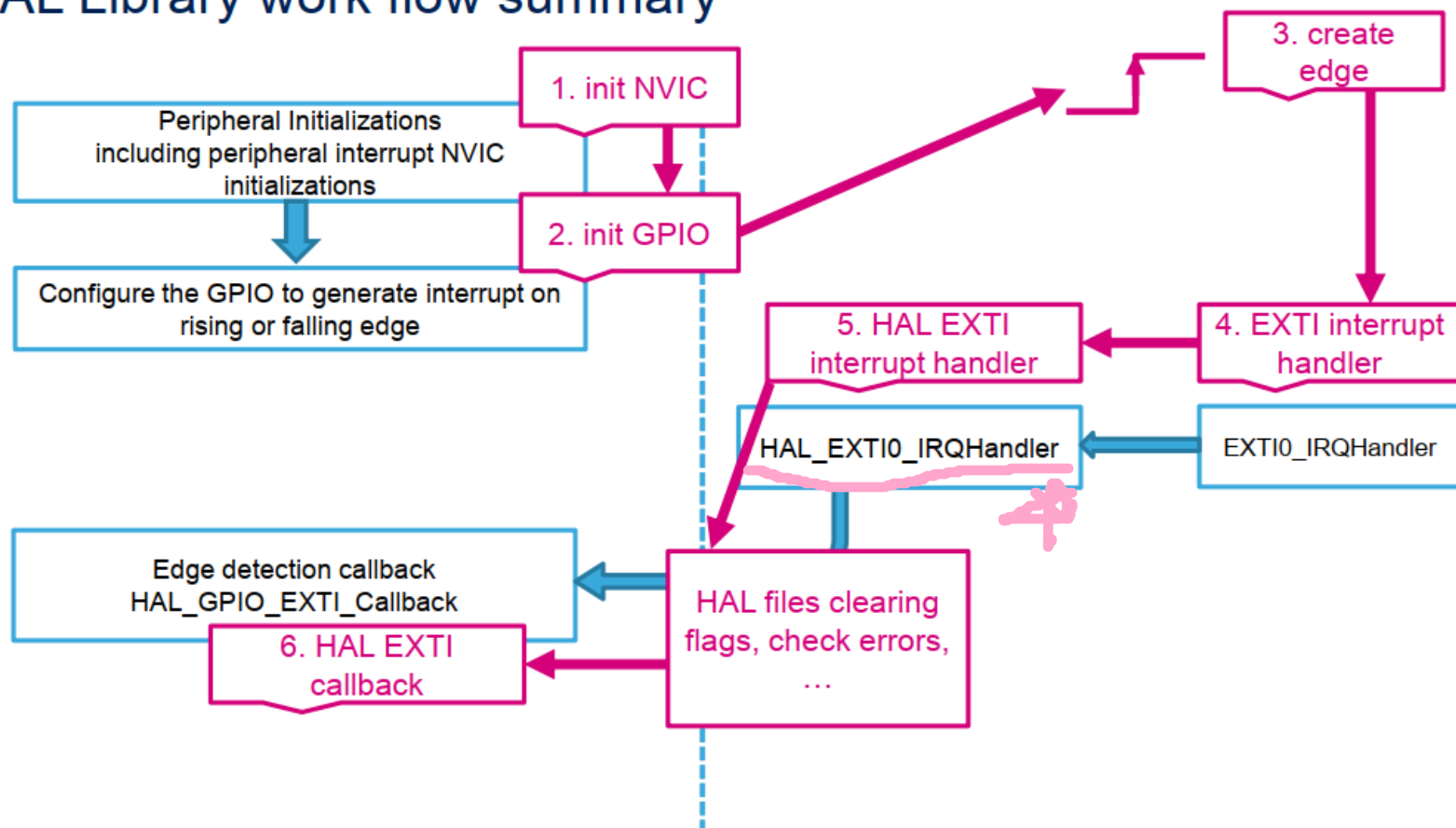
    /* EXTI interrupt init*/
    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
}
```

```
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */
    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
    /* USER CODE END EXTI15_10_IRQn 1 */
}
```

EXTI Interrupt Flow

HAL Library work flow summary



π

EXTI Interrupt Flow

```
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */
    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
    /* USER CODE END EXTI15_10_IRQn 1 */
}
```

stm32f7xx_it.c

stm32f7xx_hal_gpio.c

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
```

weak type

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        HAL_UART_Transmit(&huart3, (uint8_t *) "---", 3, 100);
        HAL_Delay(200);

        for(int i=0; i<20; i++)
        {
            HAL_UART_Transmit(&huart3, (uint8_t *) "B", 1, 100);
            HAL_Delay(200);
        }
    }
}
/* USER CODE END 4 */
```

main.c

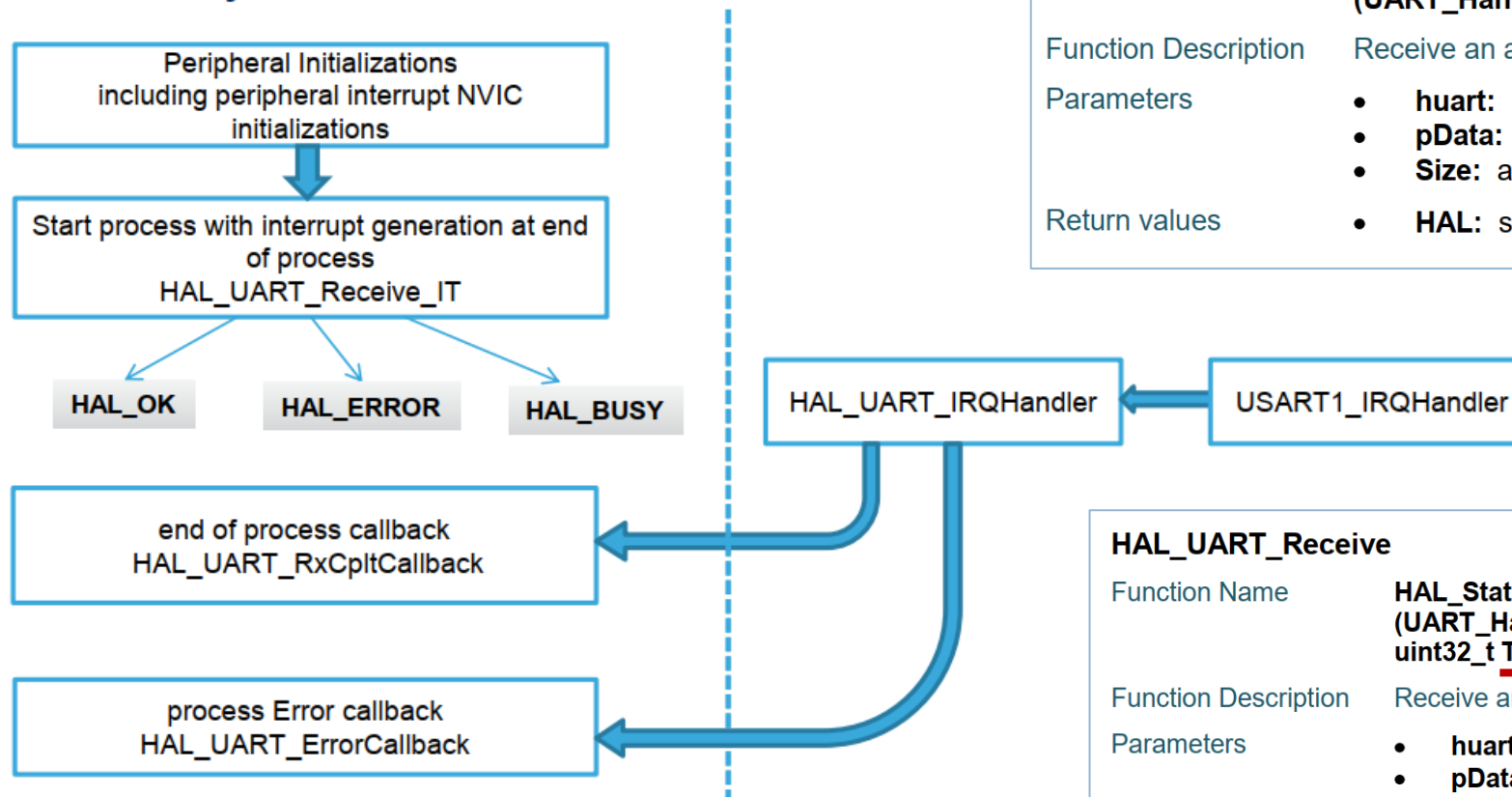
```
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);

    /* NOTE: This function Should not be modified, when the callback is needed,
    the HAL_GPIO_EXTI_Callback could be implemented in the user file */
}
```

stm32f7xx_hal_gpio.c

Receive UART Interrupt Flow

HAL Library UART with IT receive flow



HAL_UART_Receive_IT

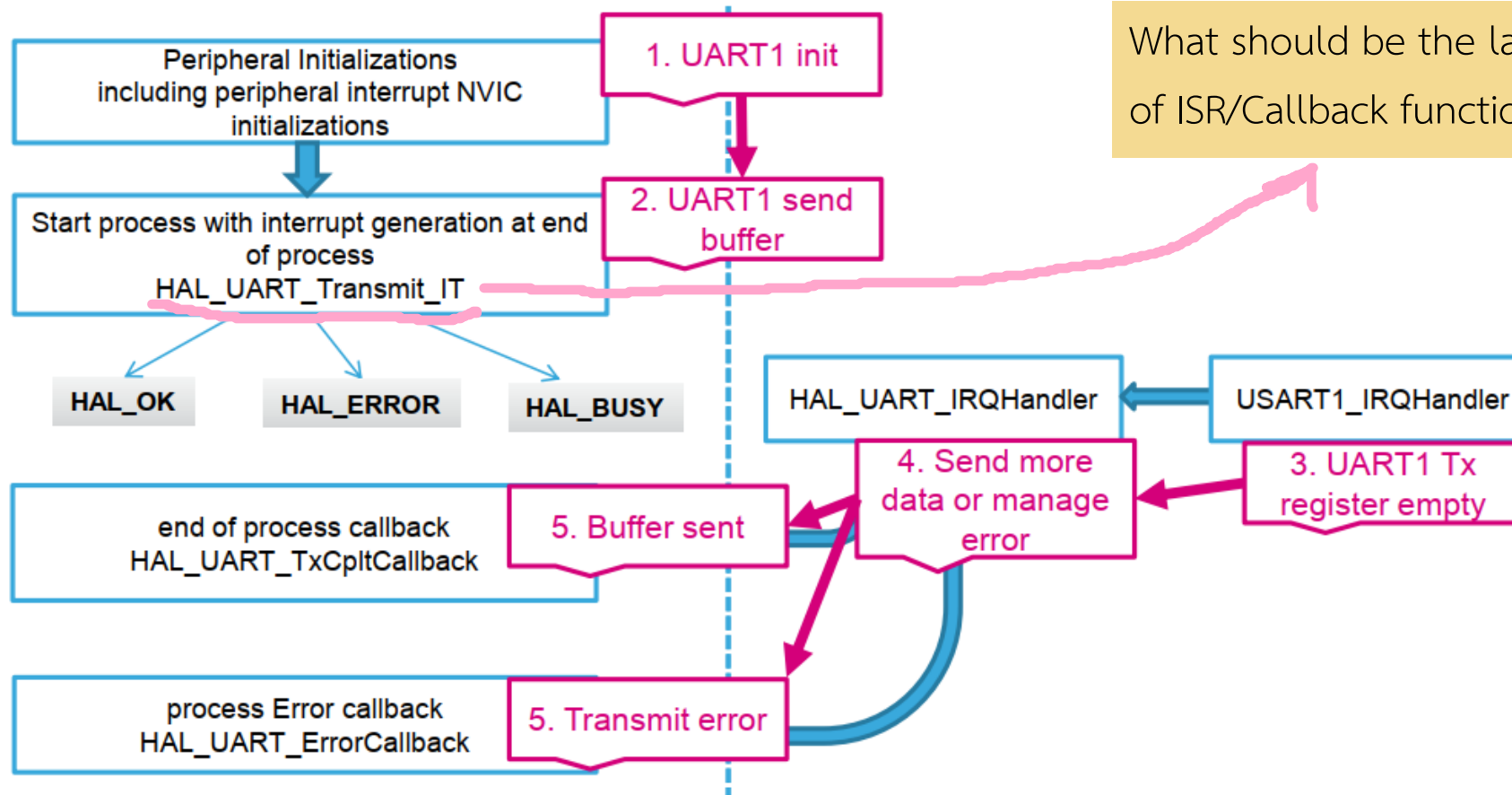
Function Name	HAL_StatusTypeDef HAL_UART_Receive_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
Function Description	Receive an amount of data in interrupt mode.
Parameters	<ul style="list-style-type: none"> huart: UART handle. pData: pointer to data buffer. Size: amount of data to be received.
Return values	<ul style="list-style-type: none"> HAL: status

HAL_UART_Receive

Function Name	HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, <u>uint32_t Timeout</u>)
Function Description	Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> huart: UART handle. pData: pointer to data buffer. Size: amount of data to be received. Timeout: Timeout duration.
Return values	<ul style="list-style-type: none"> HAL: status

Transmit UART Interrupt Flow

HAL Library UART with IT receive flow



What should be the last statement of ISR/Callback function?

Good ISR & Callback Function

➤ To Do

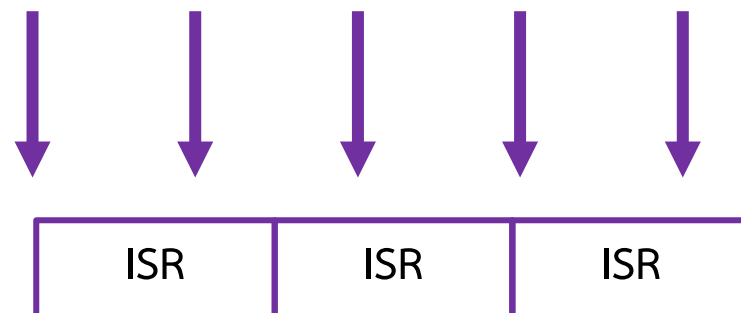
- Acknowledge Interrupt
- Clear Pending Bit
- Concise กระชับ

➤ Not To Do

- Delay loop
- Longer than Interrupt Interval

➤ Callback

- The same as ISR
- Execute after clear bit
- So less critical



Summary



- Polling is a method of repeatedly test for the change of input signals. (SW)
- Interrupt is a mechanism that responds to events or input signals by interrupting CPU
- Interrupt Service Routine is a function that automatically called after an interrupt occurs. (HW)
- Interrupt is *powerful* mechanism but can be *messy*.