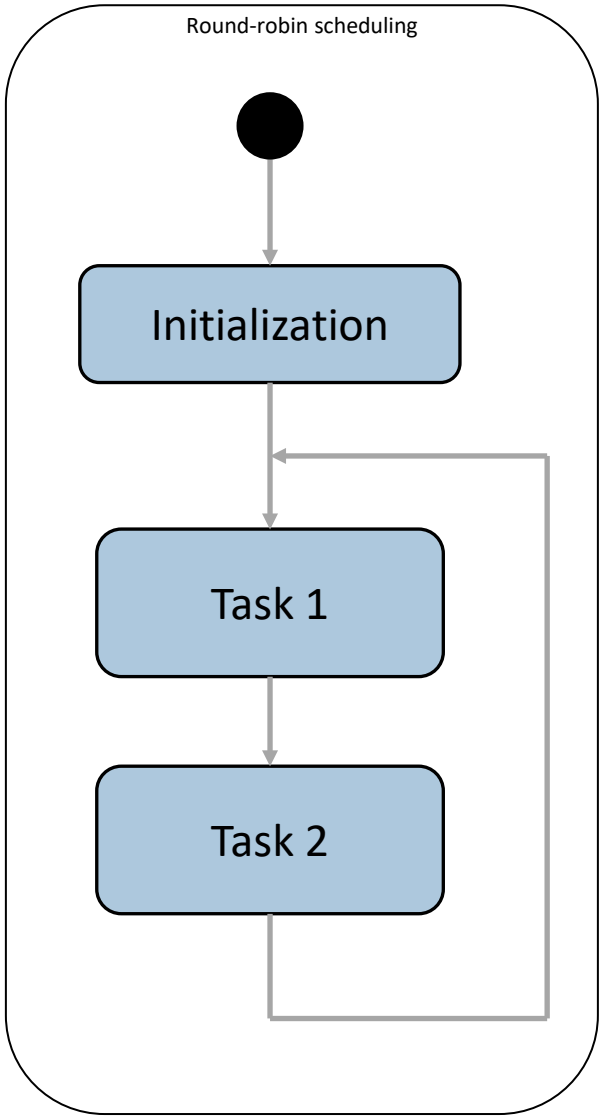
Microcomputer Engineering TMIK13 Lecture 4

INTERRUPTS

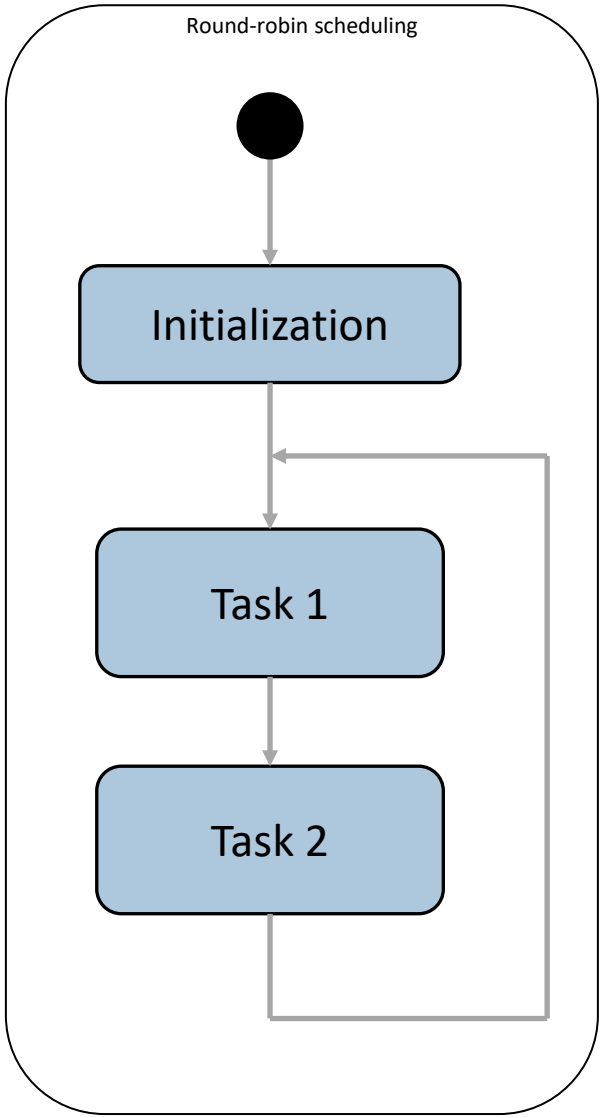
ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

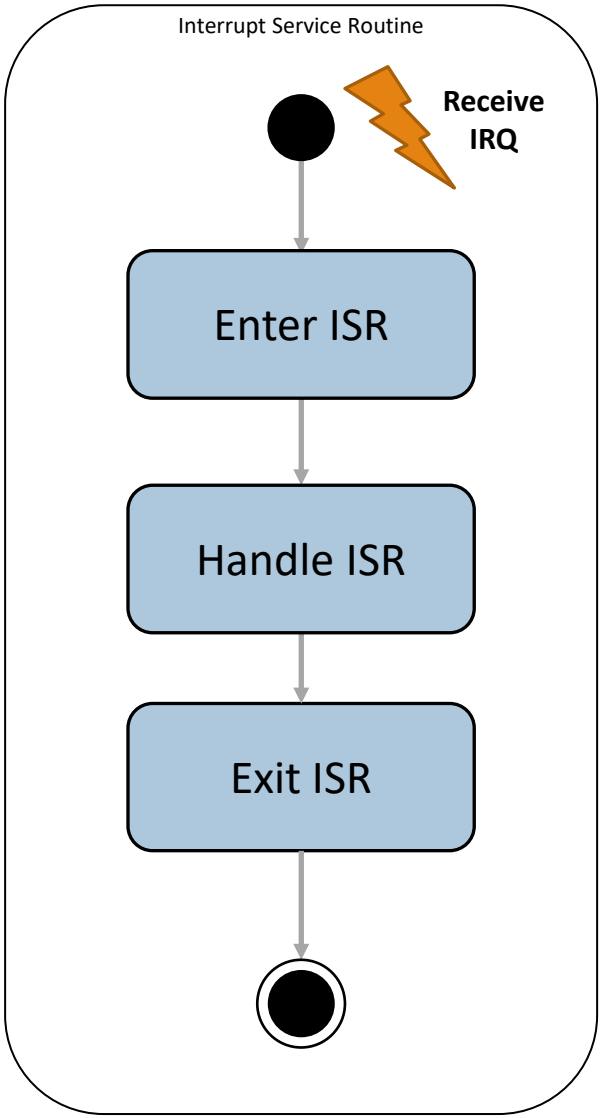
Simple Scheduling – Super Loop



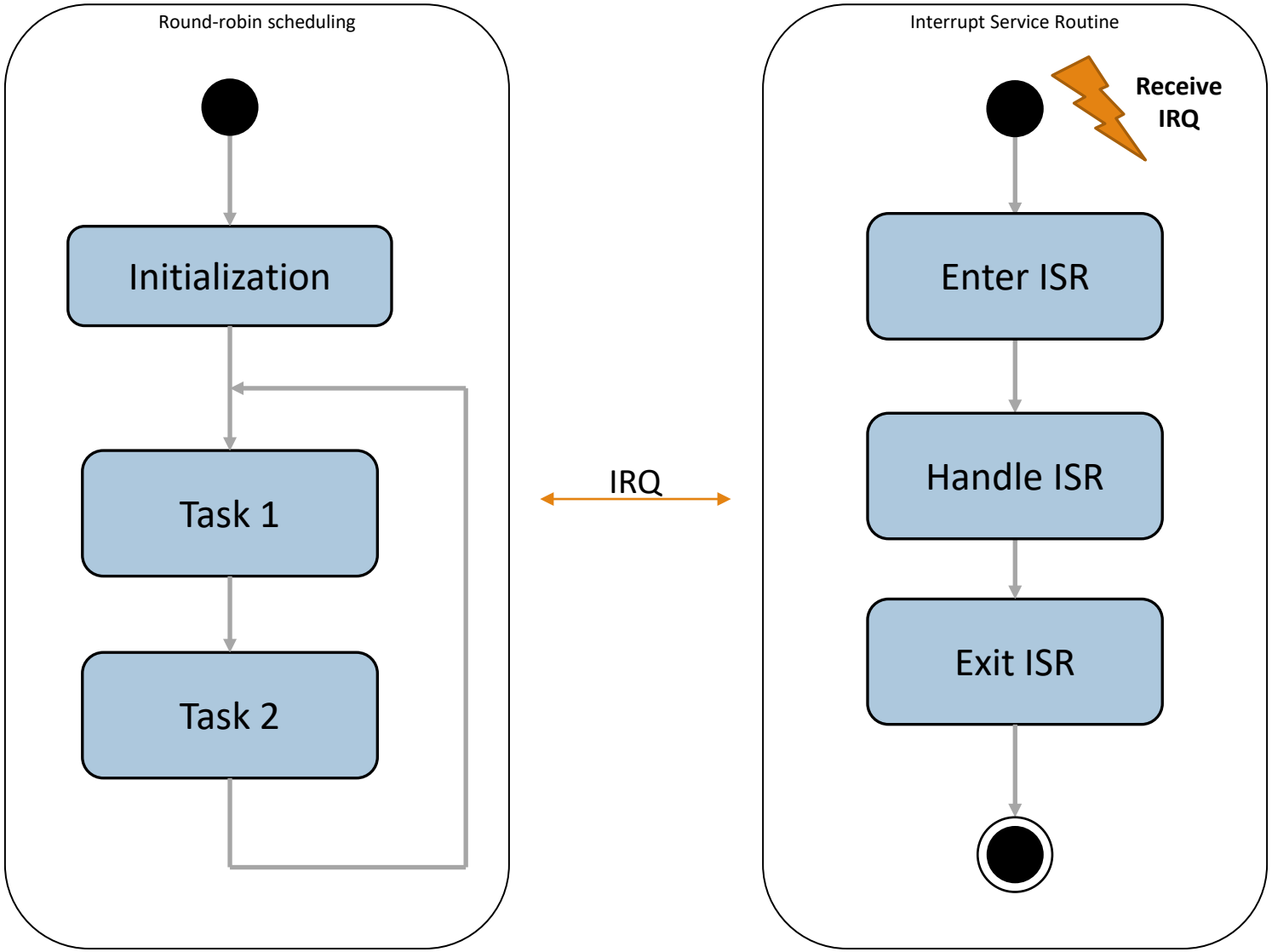

```
int main()
{
    System_Init();

    // Begin round robin scheduling
    while (1)
    {
        Task_1();

        Task_2();
    }

    return 0;
}
```

Simple Scheduling – Interrupts

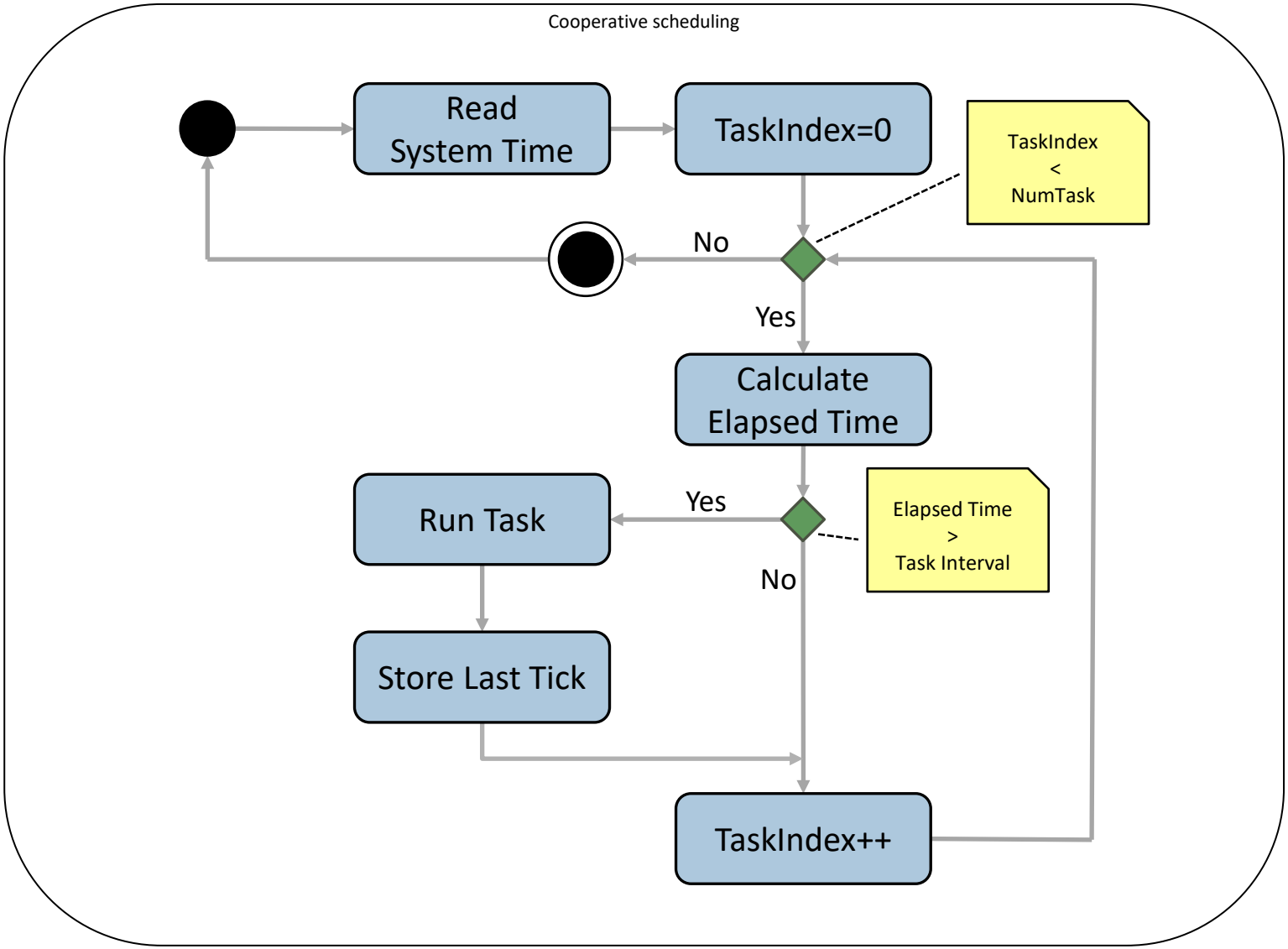




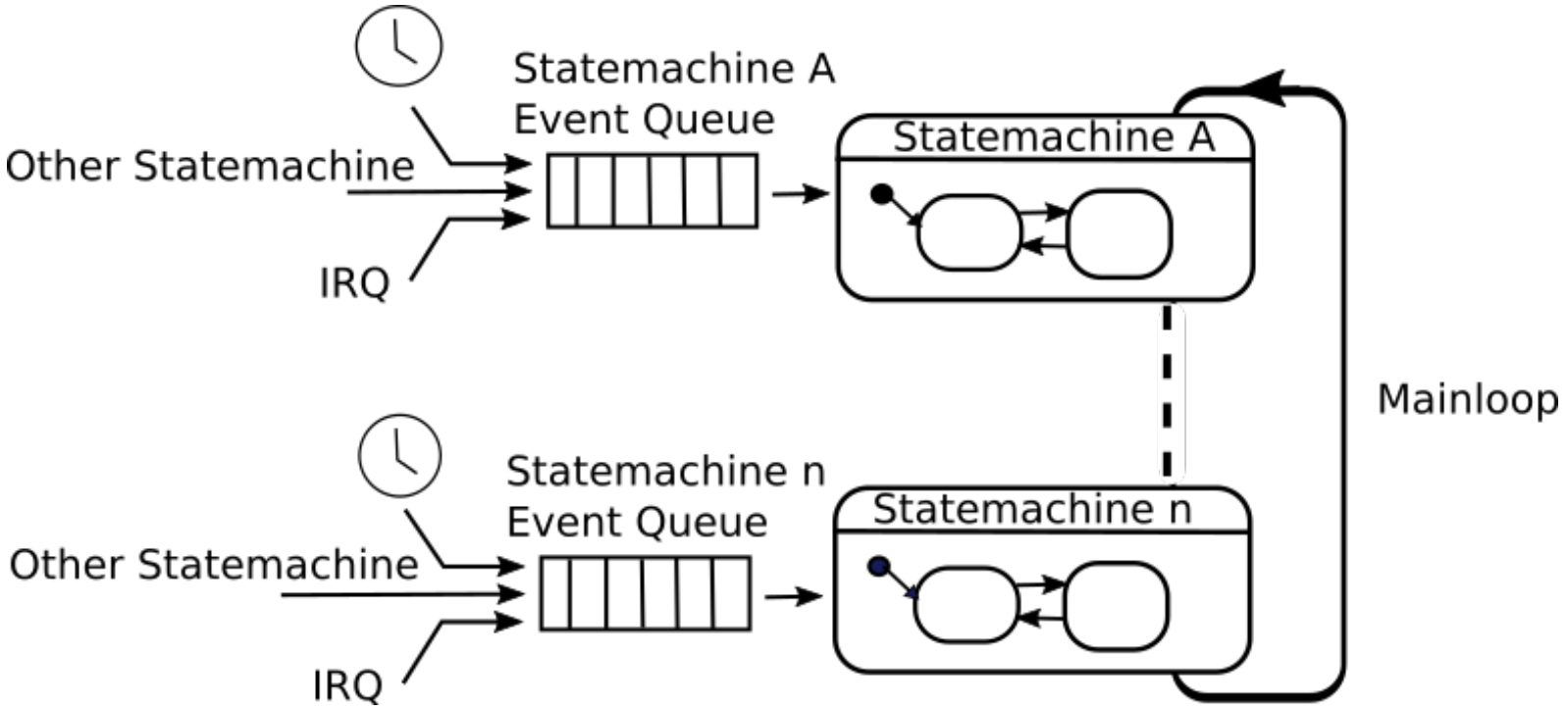
Simple Scheduling – Interrupts



Cooperative Scheduling



State Machines



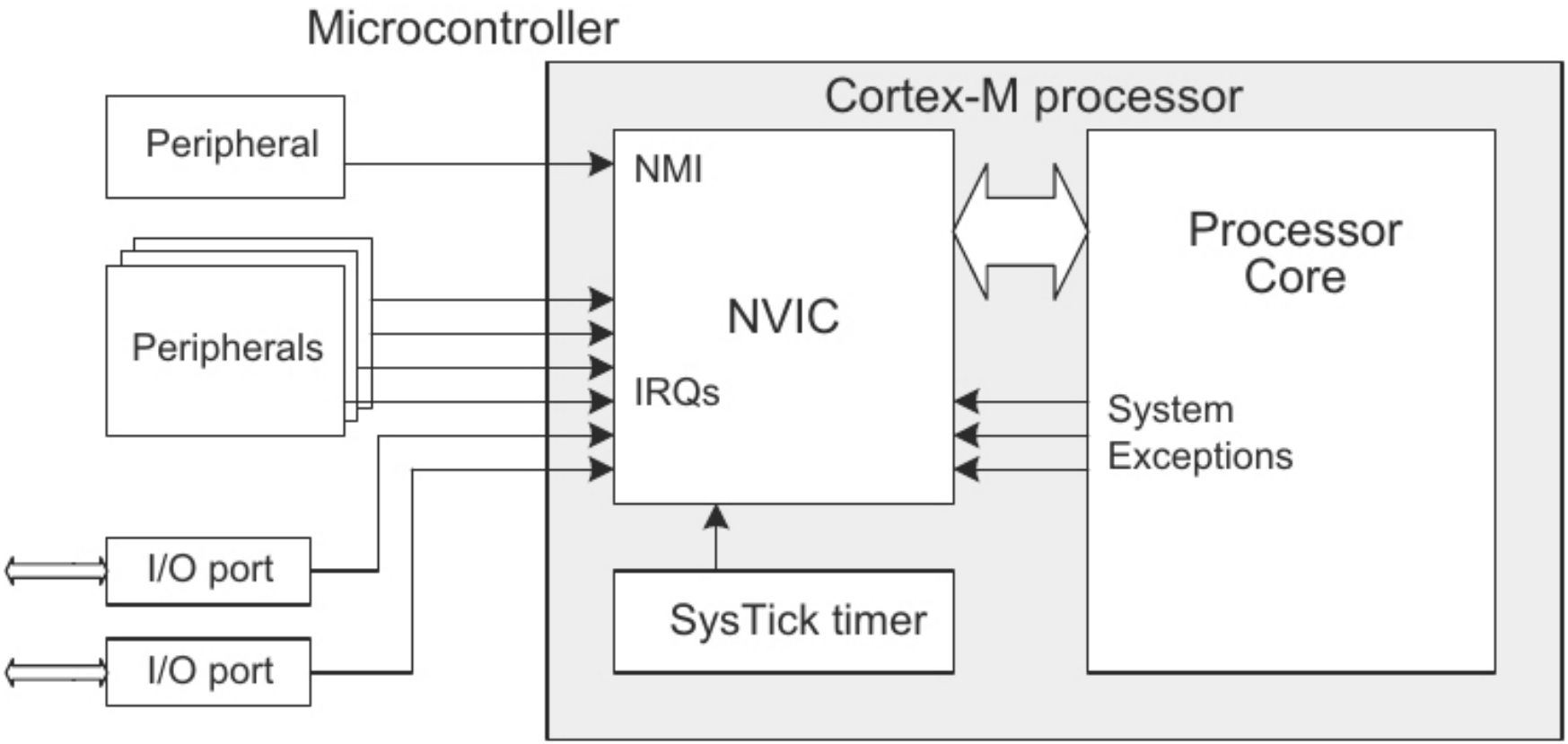
https://www.sinelabore.com/

Polling vs Interrupt

Polling

Interrupt

ARM Cortex-M Interrupts



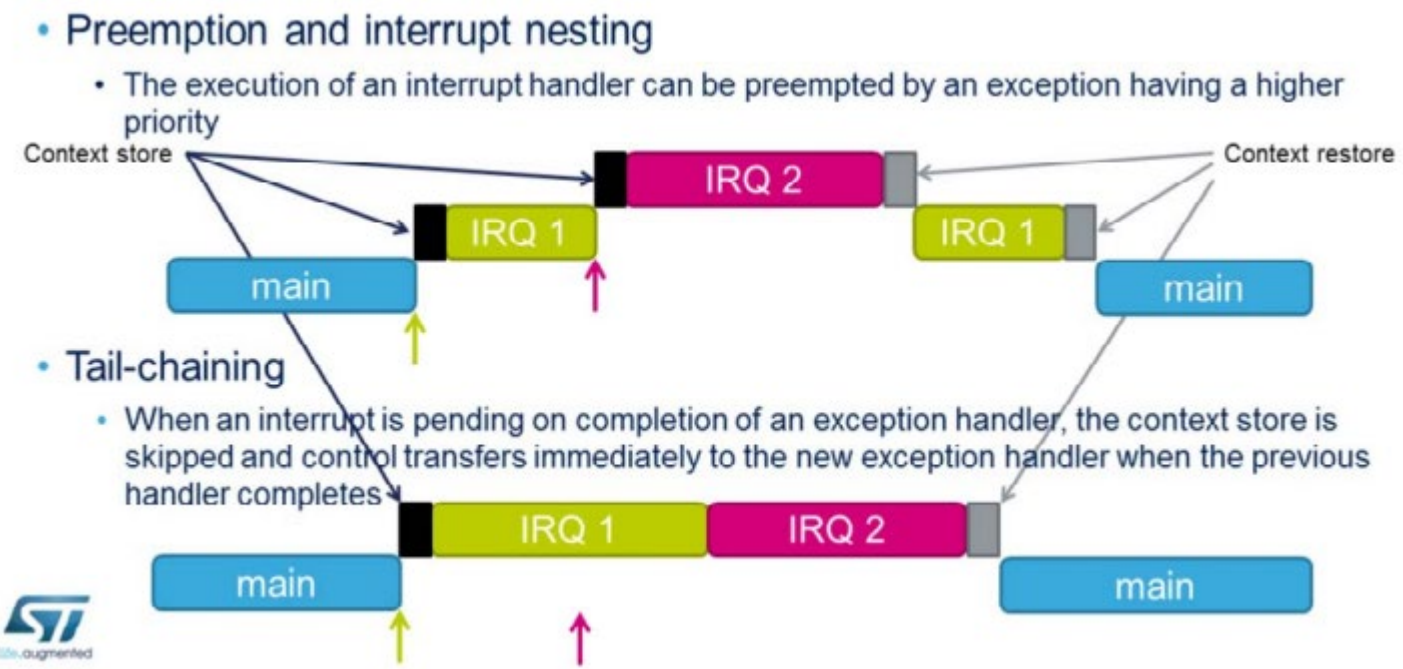
STM32F4 Interrupts – Vector table

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		SVCall
10		0x002C	Reserved
9			
8			
7			
6	-10		Usage fault
5	-11	0x0018	Bus fault
4	-12	0x0014	Memory management fault
3	-13	0x0010	Hard fault
2	-14	0x000C	NMI
1		0x0008	Reset
		0x0004	Initial SP value
		0x0000	

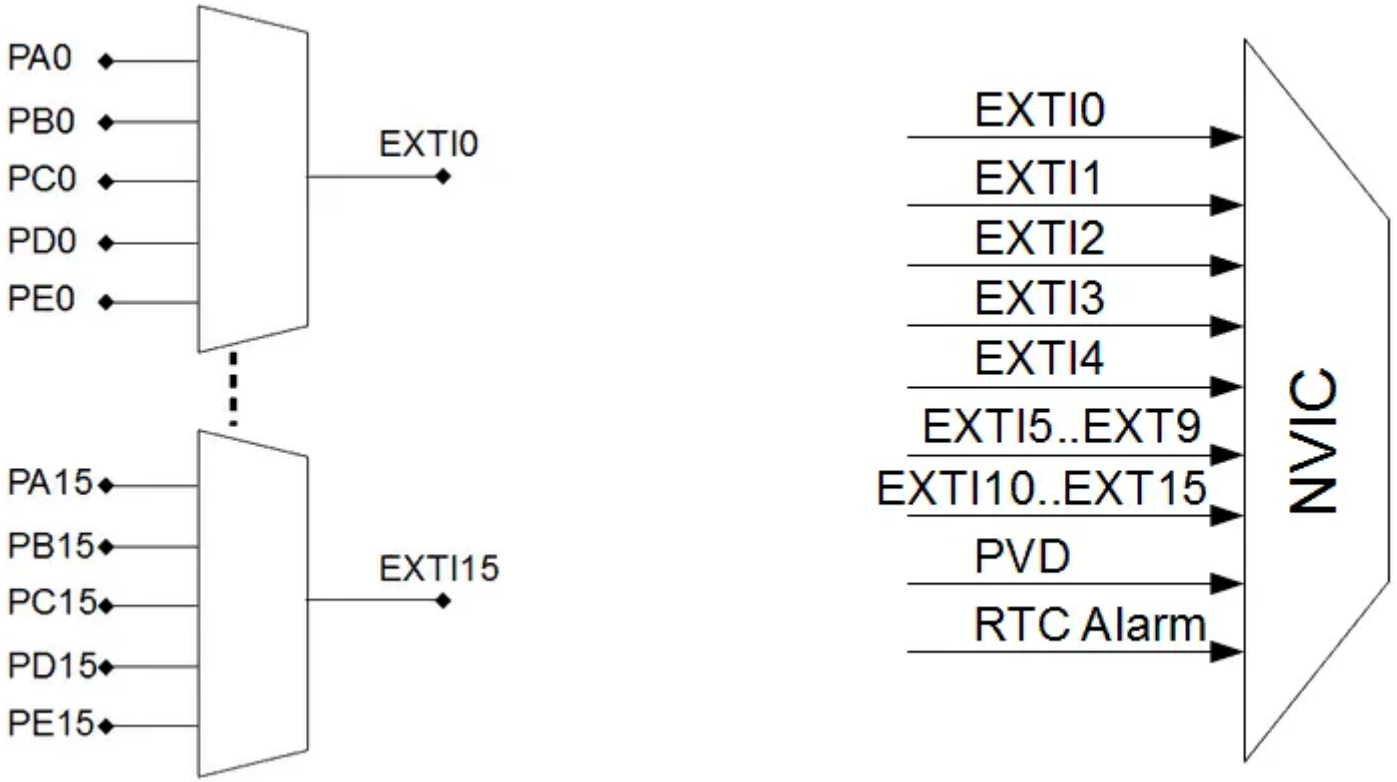
STM32F4 Interrupts – NVIC

STM32F4 contains a Nested Vectored Interrupt Controller (NVIC)

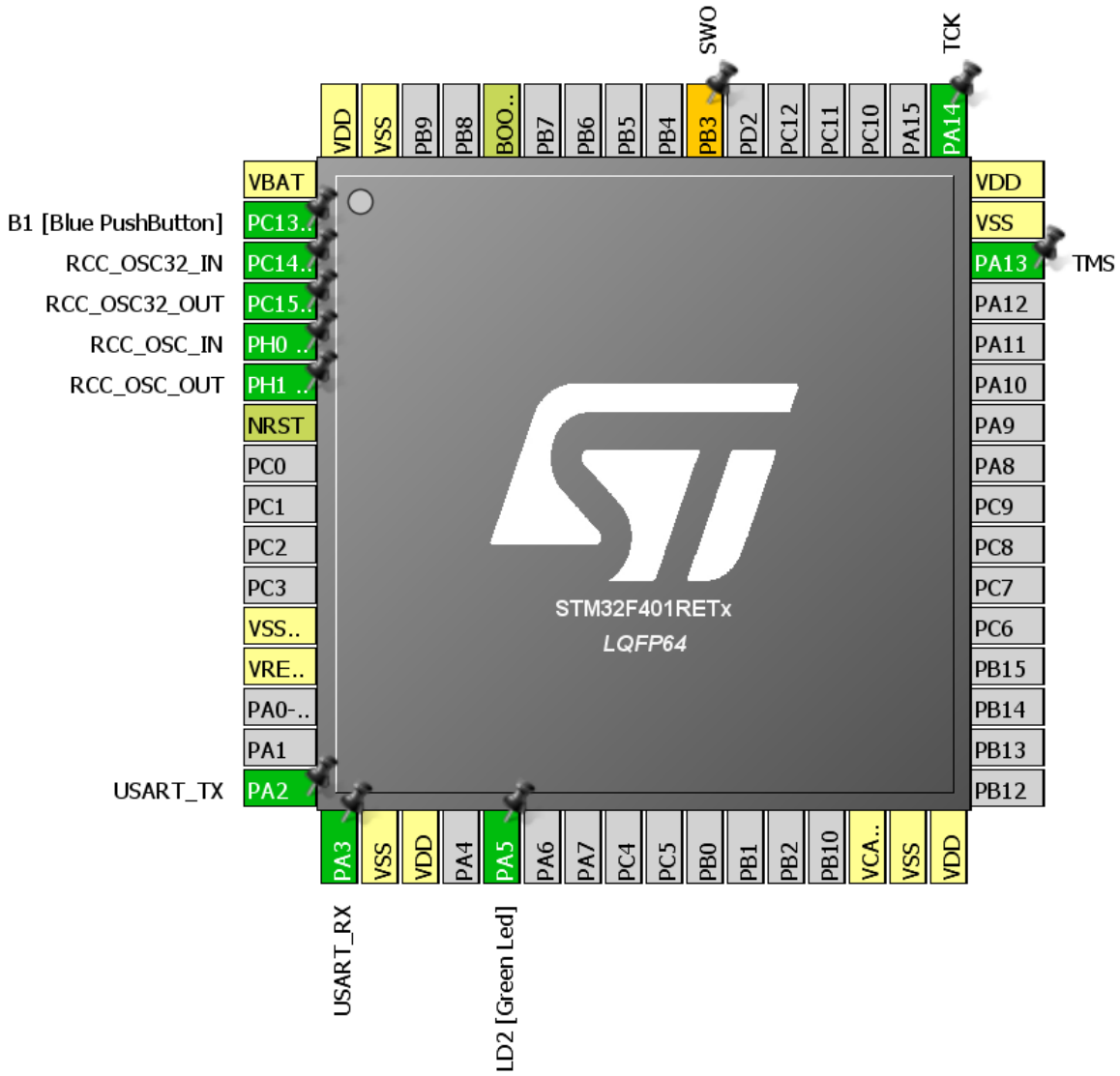
- Low latency interrupt management
- 16 programmable priority levels
- Interrupt tail chaining



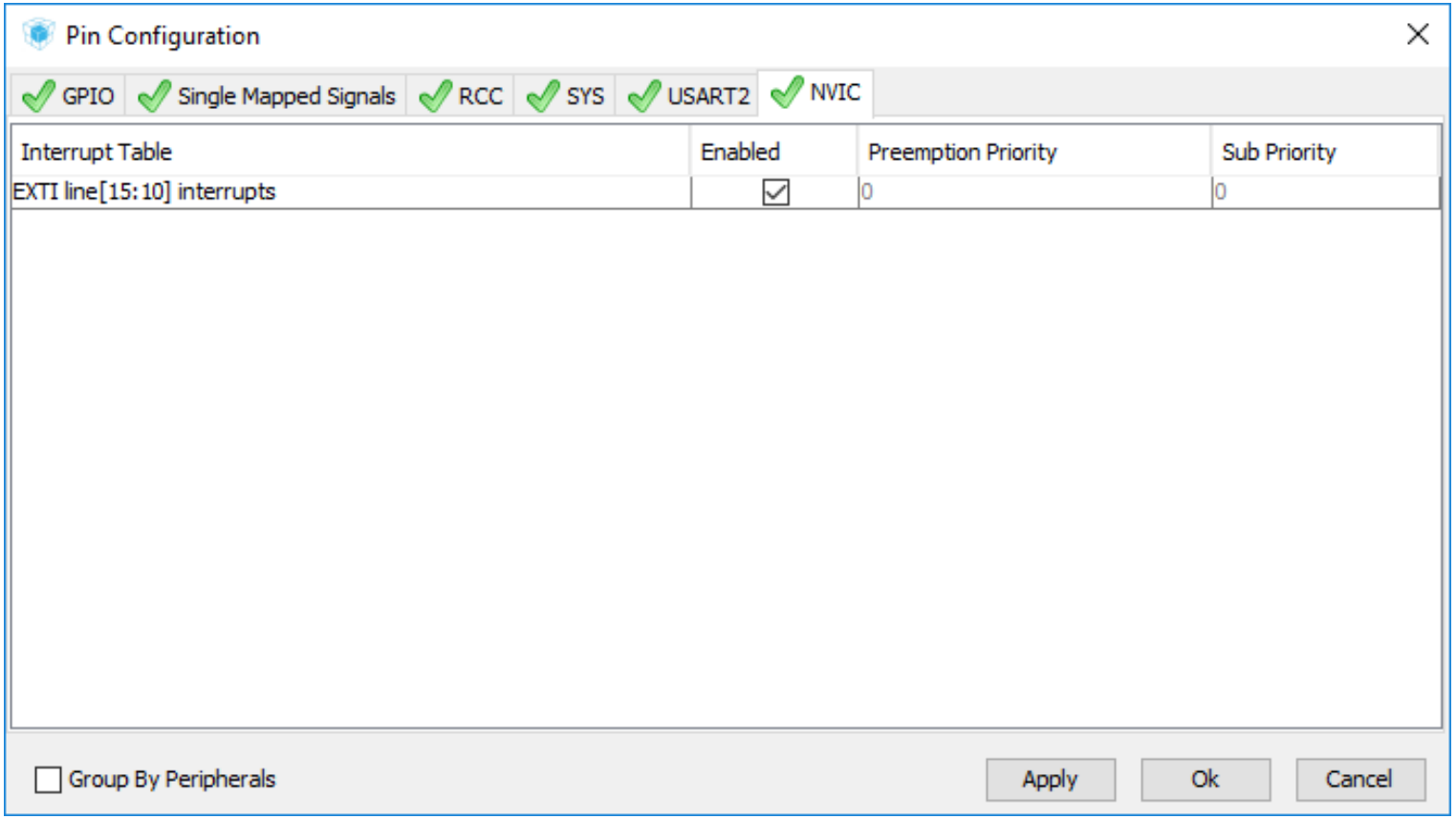
GPIO – EXTI / NVIC



STM32F4 Interrupts – GPIO



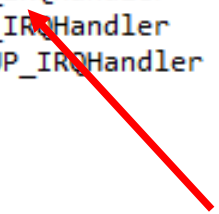
STM32F4 Interrupts – GPIO



STM32F4 Interrupts – GPIO

startup_stm32f401xe.s

```
181   .word     0                                 /* Reserved                                  */
182   .word     EXTI9_5_IRQHandler                /* External Line[9:5]s                       */
183   .word     TIM1_BRK_TIM9_IRQHandler          /* TIM1 Break and TIM9                       */
184   .word     TIM1_UP_TIM10_IRQHandler          /* TIM1 Update and TIM10                     */
185   .word     TIM1_TRG_COM_TIM11_IRQHandler     /* TIM1 Trigger and Commutation and TIM11 */
186   .word     TIM1_CC_IRQHandler                /* TIM1 Capture Compare                      */
187   .word     TIM2_IRQHandler                   /* TIM2                                      */
188   .word     TIM3_IRQHandler                   /* TIM3                                      */
189   .word     TIM4_IRQHandler                   /* TIM4                                      */
190   .word     I2C1_EV_IRQHandler                /* I2C1 Event                                */
191   .word     I2C1_ER_IRQHandler                /* I2C1 Error                                */
192   .word     I2C2_EV_IRQHandler                /* I2C2 Event                                */
193   .word     I2C2_ER_IRQHandler                /* I2C2 Error                                */
194   .word     SPI1_IRQHandler                   /* SPI1                                      */
195   .word     SPI2_IRQHandler                   /* SPI2                                      */
196   .word     USART1_IRQHandler                 /* USART1                                    */
197   .word     USART2_IRQHandler                 /* USART2                                    */
198   .word     0                                 /* Reserved                                  */
199   .word     EXTI15_10_IRQHandler              /* External Line[15:10]s                     */
200   .word     RTC_Alarm_IRQHandler              /* RTC Alarm (A and B) through EXTI Line */
201   .word     OTG_FS_WKUP_IRQHandler            /* USB OTG FS Wakeup through EXTI line */
202   .word     0                                 /* Reserved                                  */
203   .word     0                                 /* Reserved                                  */
```



STM32F4 Interrupts – GPIO

stm32f4xx_it.c

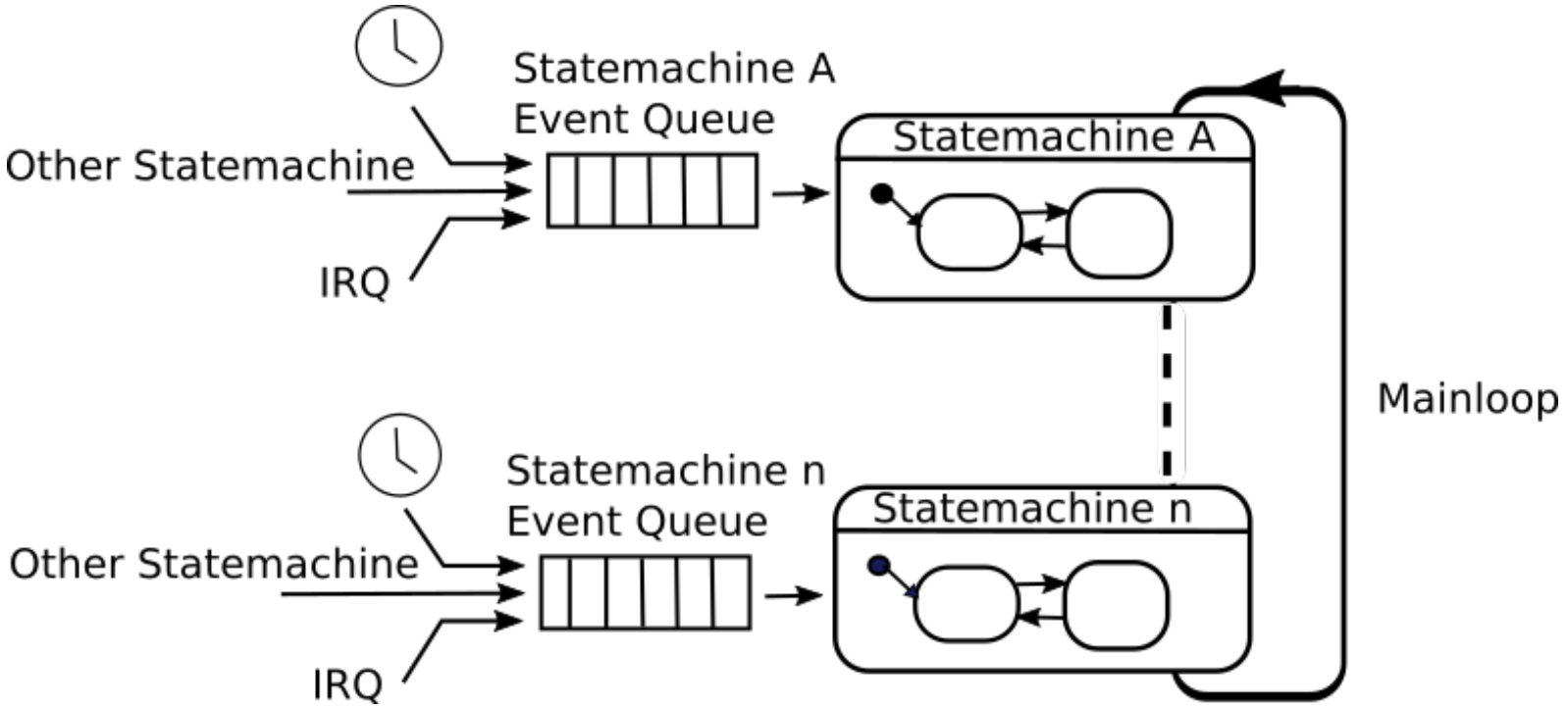
```
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
* @brief This function handles EXTI line[15:10] interrupts.
*/
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 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

STM32F4 Interrupts – GPIO

Create callback to handle GPIO interrupt

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
 if (GPIO_Pin == GPIO_PIN_13)
 {
 /* DO the job when EXTI was detected on pin 13 */
 }
}
```

State Machines

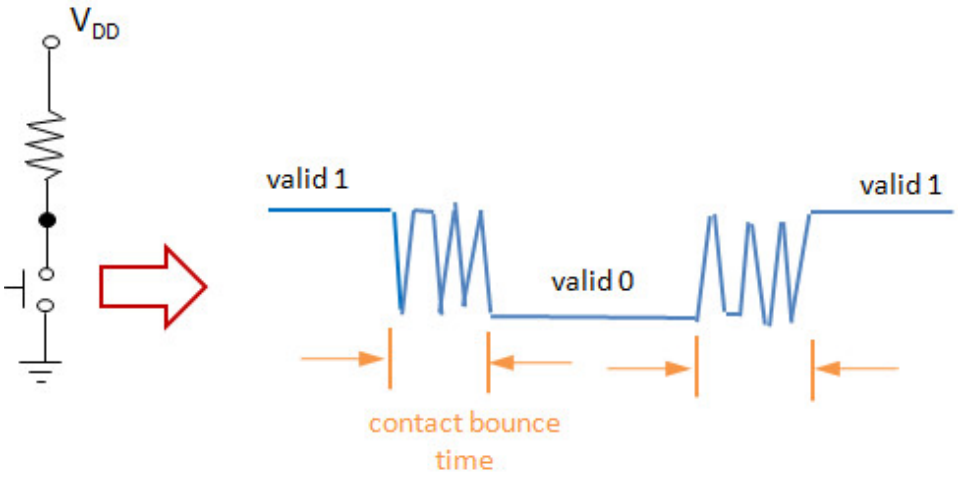



```
{
   __disable_irq();
      Put New event into event queue
   __enable_irq();
}
```

https://www.sinelabore.com/

STM32F4 Interrupts – GPIO

What about contact bounces???



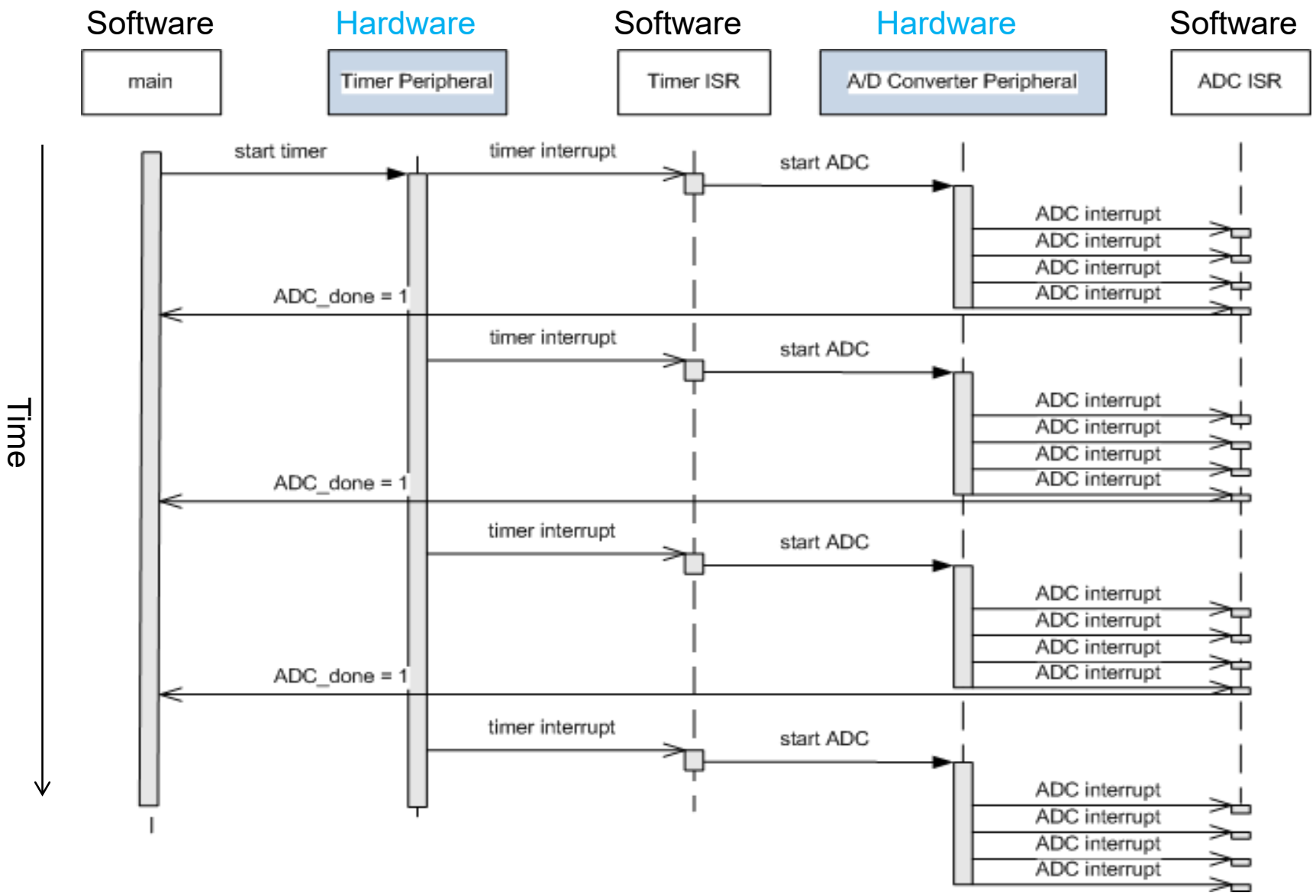
STM32F4 Interrupts

Interrupts can be generated for:

- External pin changes (EXTI)
- Timer events (Timer overflow etc)
- USART
- SPI
- DMA
- Etc

Correctly implemented these events can drive a state machine

Concurrent HW and SW operation



Microcomputer Engineering

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760