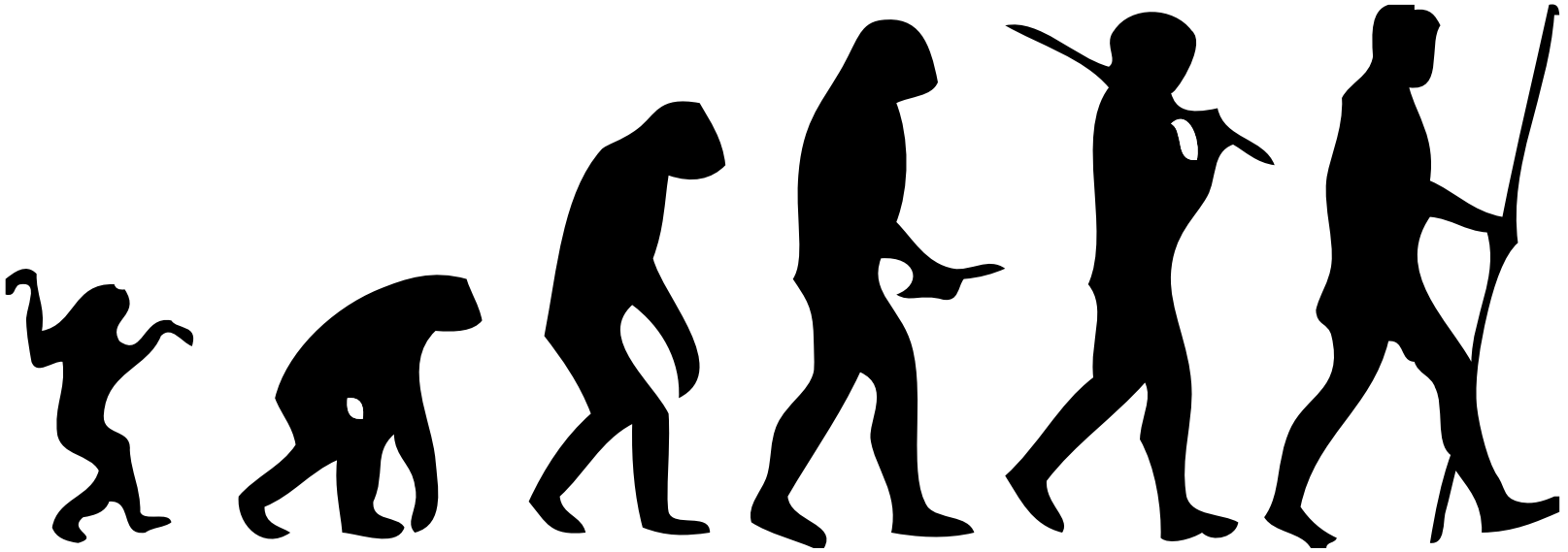
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

DISCA

Interrupciones en los microcontroladores St STM32



2012/07/07

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

DISCA

Contenido

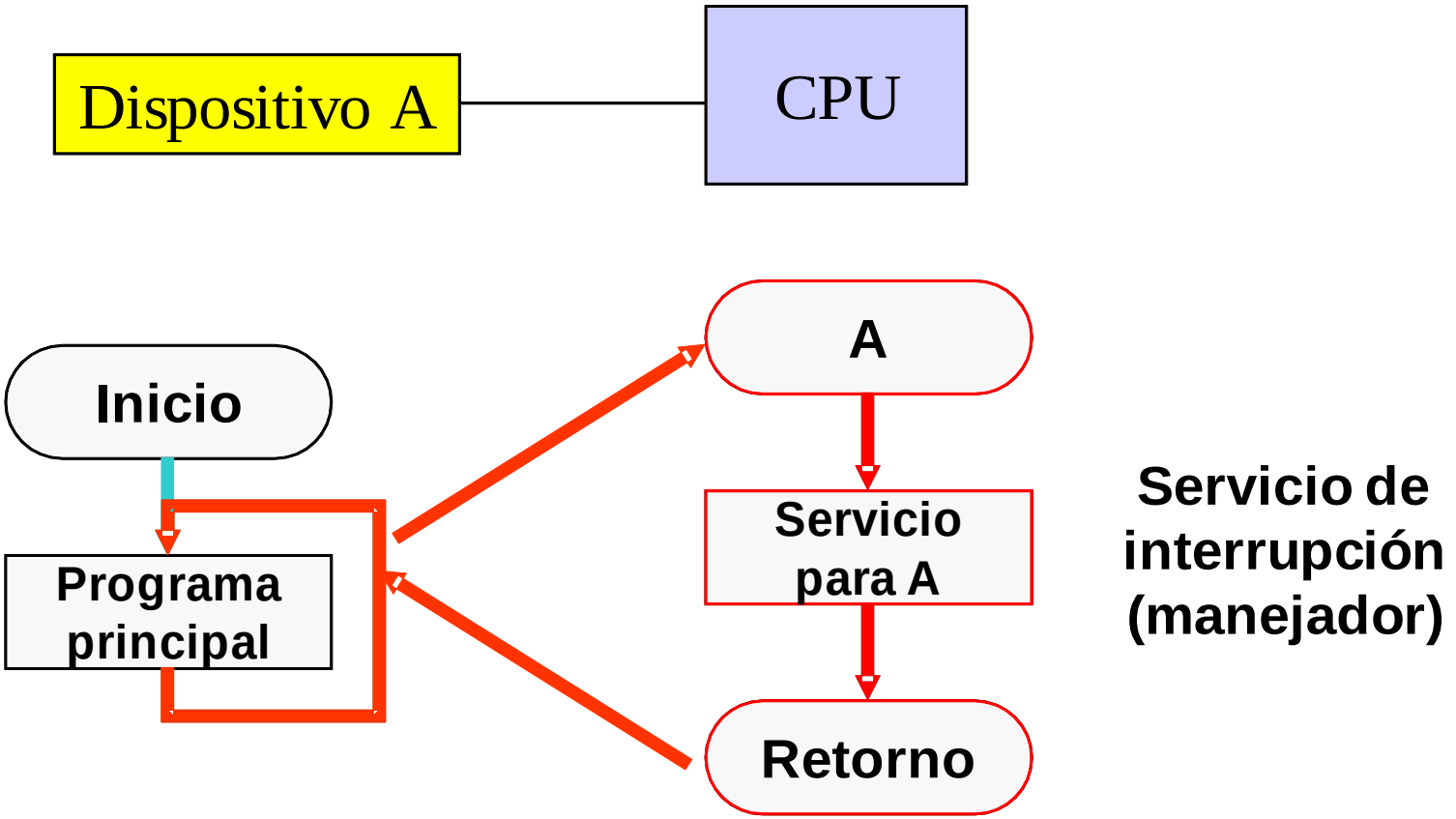
- Objetivos
- Introducción. Concepto de interrupción. Tipos
- Interrupciones en ARM Cortex-M
- Gestión de las interrupciones
- Interrupciones externas
- Ejemplo: EXTI
- Ejercicios

Objetivos

- Conocer el concepto de interrupción.
- Diferenciar las fuentes de interrupción.
- Diseñar aplicaciones mediante la utilización de interrupciones.

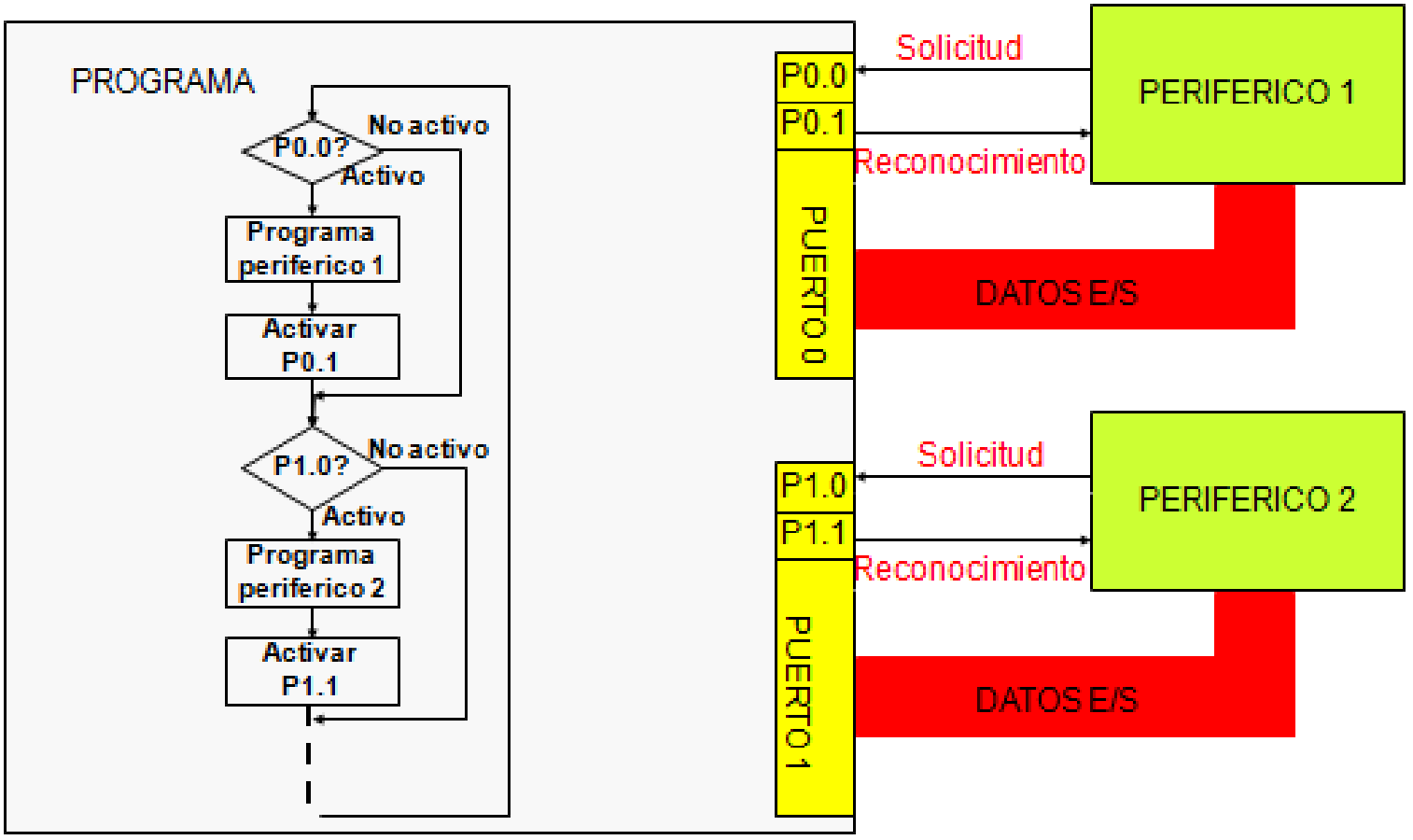
Introducción. Concepto de interrupción

- El programa en curso, tras un requerimiento, paraliza su ejecución y pasa a ejecutarse una rutina (*manejador*) de interrupción.



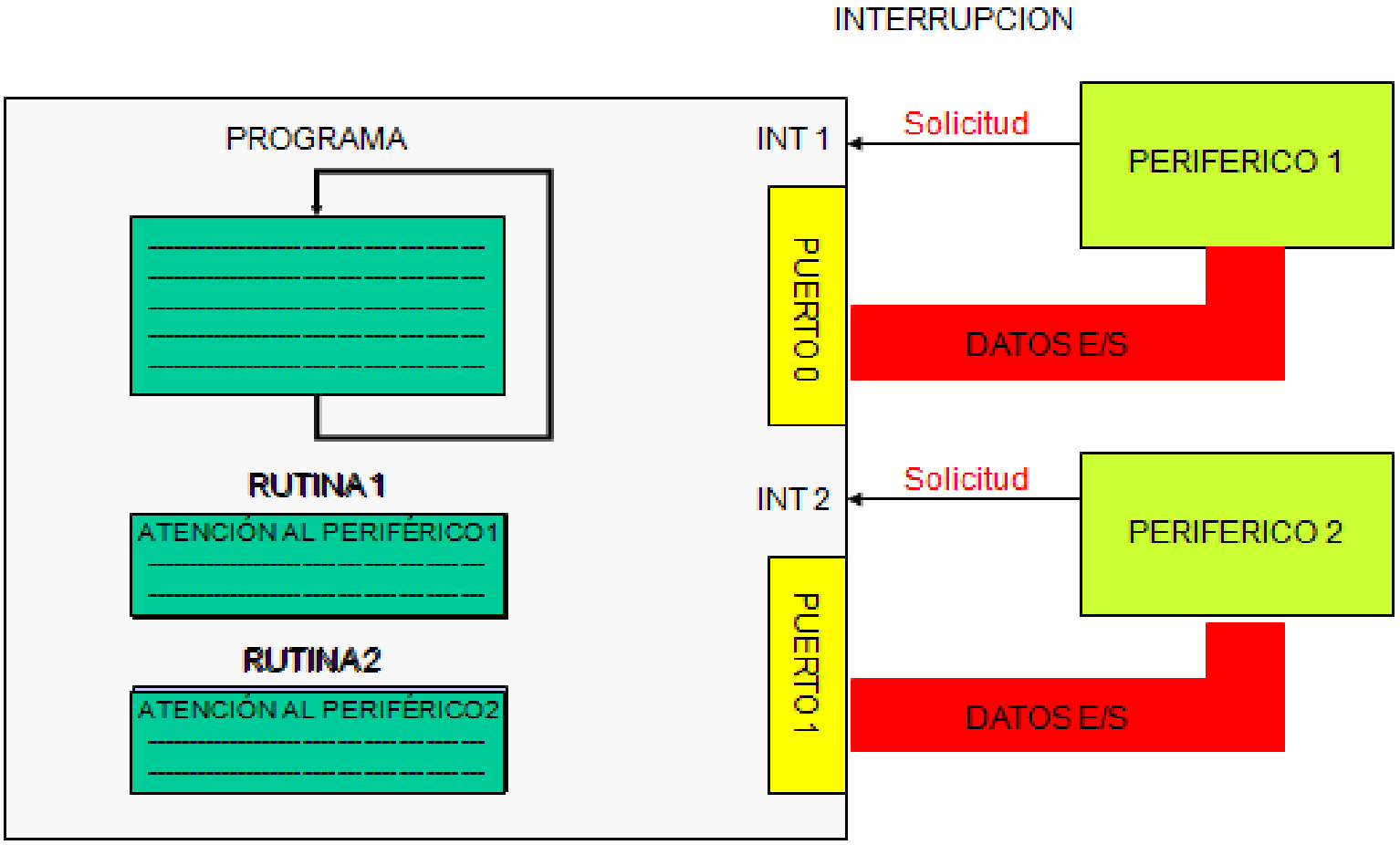
Introducción

- Atención de periféricos mediante consulta de estado (***POLLING***)



Introducción

- Atención de periféricos mediante **INTERRUPCIONES**



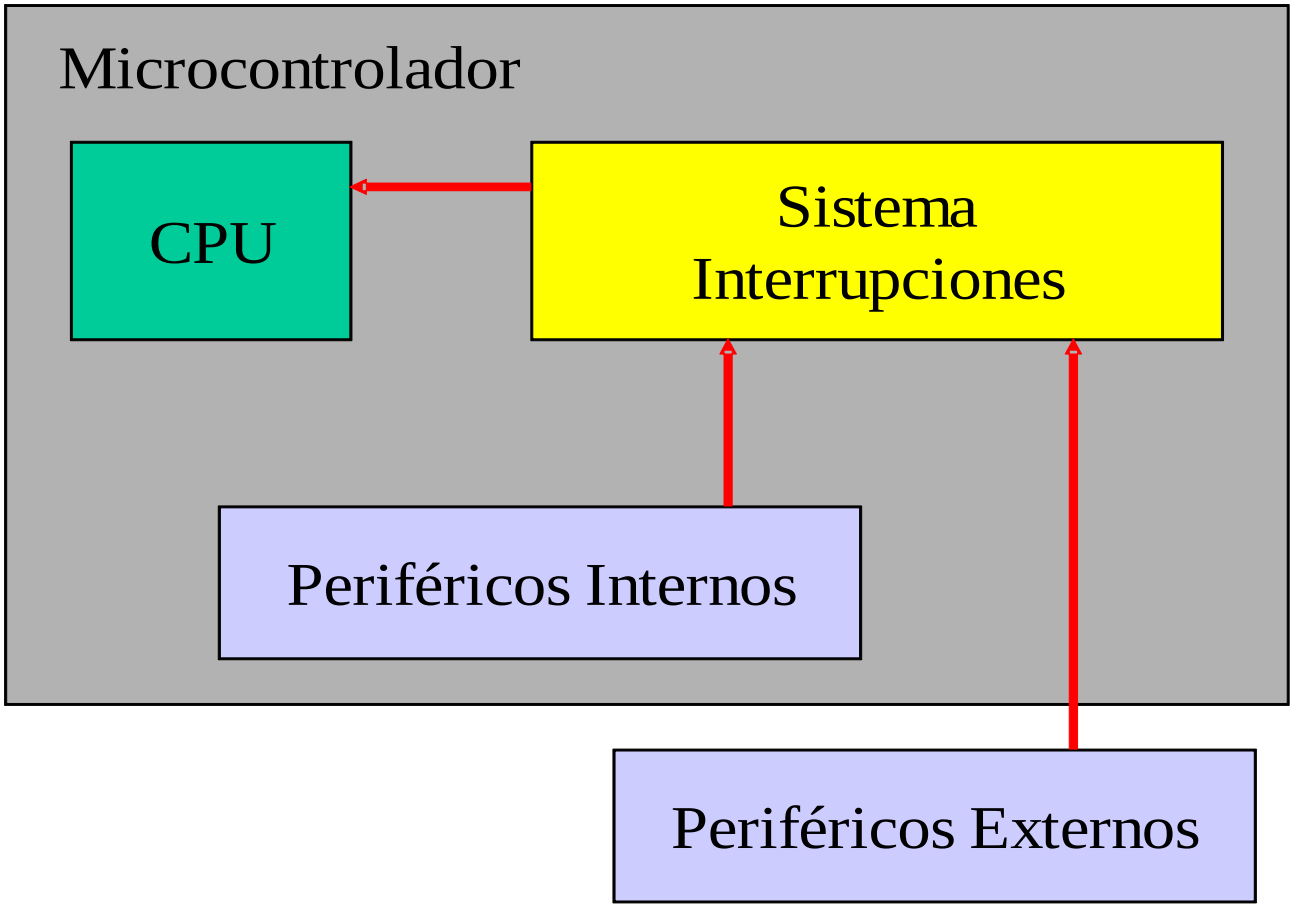
Interrupciones en ARM Cortex-M

- Soporta hasta 240 fuentes de interrupción con 256 niveles de prioridad. (M4)

- Las que nos interesan ahora son:
  - Generadas por fuentes externas.
    - EXTI0,1,2,..
    - La petición de interrupción se puede generar tanto por nivel como por flanco en el pin correspondiente
  - Interrupciones generadas por los periféricos internos.
    - Temporizadores/Contadores

- Cada fuente de interrupción tiene asociado su propio vector de interrupción para localizar el manejador.

Interrupciones en ARM Cortex-M



Interrupciones en ARM Cortex-M

- Los ARM Cortex-M disponen de un controlador de interrupciones: Nested Vectored Interrupt Controller (NVIC)
- Cuando se produce una interrupción el NVIC compara la prioridad de esta interrupción (P_i) con el nivel de prioridad de ejecución actual (P_a)
  - Si $P_i > P_a$ se ejecuta el manejador de la interrupción
- Las interrupciones puede ser habilitadas / inhibidas
- Los micros Cortex-M incluyen características avanzadas para reducir la latencia de la interrupción

Gestión de las interrupciones

- CMSIS HAL (*Hardware Abstraction Layer*) utiliza números de IRQ (IRQn) para identificar las interrupciones.
  - La primera interrupción de dispositivo tiene el IRQn = 0
  - Se utilizan valores negativos de IRQn para las "*processor core exceptions*"

- El fichero stm32f4xx.h lo proporciona el fabricante del micro y es el principal *include* en las aplicaciones..
- Este fichero contiene entre otras cosas:
  - *Interrupt Number Definition*: proporciona los números de interrupción (IRQn) para todas las interrupciones y excepciones.

Gestión de las interrupciones

- stm32f4xx.h

```
/**
 * @brief STM32F4XX Interrupt Number Definition, according to the selected device
 *        in @ref Library_configuration_section
 */
typedef enum IRQn
{
/******  Cortex-M4 Processor Exceptions Numbers ***********************************************************************/
  NonMaskableInt_IRQn         = -14,    /*!< 2 Non Maskable Interrupt                                                */
  MemoryManagement_IRQn       = -12,    /*!< 4 Cortex-M4 Memory Management Interrupt                                 */
  BusFault_IRQn               = -11,    /*!< 5 Cortex-M4 Bus Fault Interrupt                                         */
  UsageFault_IRQn             = -10,    /*!< 6 Cortex-M4 Usage Fault Interrupt                                       */
  SVCall_IRQn                 = -5,     /*!< 11 Cortex-M4 SV Call Interrupt                                          */
  DebugMonitor_IRQn           = -4,     /*!< 12 Cortex-M4 Debug Monitor Interrupt                                    */
  PendSV_IRQn                 = -2,     /*!< 14 Cortex-M4 Pend SV Interrupt                                          */
  SysTick_IRQn                = -1,     /*!< 15 Cortex-M4 System Tick Interrupt                                      */
/******  STM32 specific Interrupt Numbers *****************************************************************************/
  WWDG_IRQn                   = 0,      /*!< Window WatchDog Interrupt                                               */
  PVD_IRQn                    = 1,      /*!< PVD through EXTI Line detection Interrupt                               */
  TAMP_STAMP_IRQn             = 2,      /*!< Tamper and TimeStamp interrupts through the EXTI line                   */
  RTC_WKUP_IRQn               = 3,      /*!< RTC Wakeup interrupt through the EXTI line                              */
  FLASH_IRQn                  = 4,      /*!< FLASH global Interrupt                                                  */
  RCC_IRQn                    = 5,      /*!< RCC global Interrupt                                                    */
  EXTI0_IRQn                  = 6,      /*!< EXTI Line0 Interrupt                                                    */
  EXTI1_IRQn                  = 7,      /*!< EXTI Line1 Interrupt                                                    */
  EXTI2_IRQn                  = 8,      /*!< EXTI Line2 Interrupt                                                    */
  EXTI3_IRQn                  = 9,      /*!< EXTI Line3 Interrupt                                                    */
  EXTI4_IRQn                  = 10,     /*!< EXTI Line4 Interrupt                                                    */
  DMA1_Stream0_IRQn           = 11,     /*!< DMA1 Stream 0 global Interrupt                                          */
  DMA1_Stream1_IRQn           = 12,     /*!< DMA1 Stream 1 global Interrupt                                          */
  DMA1_Stream2_IRQn           = 13,     /*!< DMA1 Stream 2 global Interrupt                                          */
  DMA1_Stream3_IRQn           = 14,     /*!< DMA1 Stream 3 global Interrupt                                          */
  DMA1_Stream4_IRQn           = 15,     /*!< DMA1 Stream 4 global Interrupt                                          */
  DMA1_Stream5_IRQn           = 16,     /*!< DMA1 Stream 5 global Interrupt                                          */
  DMA1_Stream6_IRQn           = 17,     /*!< DMA1 Stream 6 global Interrupt                                          */
  ADC_IRQn                    = 18,     /*!< ADC1, ADC2 and ADC3 global Interrupts                                   */
  CAN1_TX_IRQn                = 19,     /*!< CAN1 TX Interrupt                                                       */
  CAN1_RX0_IRQn               = 20,     /*!< CAN1 RX0 Interrupt                                                      */
  CAN1_RX1_IRQn               = 21,     /*!< CAN1 RX1 Interrupt                                                      */
  CAN1_SCE_IRQn               = 22,     /*!< CAN1 SCE Interrupt                                                      */
```

...

Gestión de las interrupciones

- ARM proporciona una plantilla del fichero *startup_device* para cada compilador soportado
  - Este fichero es adaptado por el fabricante del micro para definir las excepciones e interrupciones así como vectores de interrupción para todos manejadores de interrupción.
    - En nuestro caso: *startup_stm32f4xx.s*
  - Cada manejador (*handler*) se define como una función *weak*
    - *De esta forma los manejadores de interrupción se pueden implementar directamente en la aplicación sin necesidad de adaptar el fichero startup*.

Gestión de las interrupciones

- startup_stm32f4xx.s
  - Los siguientes nombres de excepciones e interrupciones se definen al inicio del vector para un Cortex-M4:

```
__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     MemManage_Handler          ; MPU Fault Handler
                DCD     BusFault_Handler           ; Bus Fault Handler
                DCD     UsageFault_Handler         ; Usage Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     SVC_Handler                ; SVCall Handler
                DCD     DebugMon_Handler           ; Debug Monitor Handler
                DCD     0                          ; Reserved
                DCD     PendSV_Handler             ; PendSV Handler
                DCD     SysTick_Handler            ; SysTick Handler

                ; External Interrupts
                DCD     WWDG_IRQHandler                   ; Window WatchDog
                DCD     PVD_IRQHandler                    ; PVD through EXTI Line detection
                DCD     TAMP_STAMP_IRQHandler             ; Tamper and TimeStamps through the EXTI line
                DCD     RTC_WKUP_IRQHandler               ; RTC Wakeup through the EXTI line
                DCD     FLASH_IRQHandler                  ; FLASH
                DCD     RCC_IRQHandler                    ; RCC
                DCD     EXTI0_IRQHandler                  ; EXTI Line0
                DCD     EXTI1_IRQHandler                  ; EXTI Line1
                DCD     EXTI2_IRQHandler                  ; EXTI Line2
```

Gestión de las interrupciones

- startup_stm32f4xx.s
  - Las interrupciones deben tener una función ficticia que se puede sobrescribir en la aplicación de usuario:

```
Default_Handler PROC

                EXPORT  WWDG_IRQHandler                 [WEAK]
                EXPORT  PVD_IRQHandler                  [WEAK]
                EXPORT  TAMP_STAMP_IRQHandler           [WEAK]
                EXPORT  RTC_WKUP_IRQHandler             [WEAK]
                EXPORT  FLASH_IRQHandler                [WEAK]
                EXPORT  RCC_IRQHandler                  [WEAK]
                EXPORT  EXTI0_IRQHandler                [WEAK]
                EXPORT  EXTI1_IRQHandler                [WEAK]
                EXPORT  EXTI2_IRQHandler                [WEAK]
                EXPORT  EXTI3_IRQHandler                [WEAK]
                EXPORT  EXTI4_IRQHandler                [WEAK]
                EXPORT  DMA1_Stream0_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream1_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream2_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream3_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream4_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream5_IRQHandler         [WEAK]
                EXPORT  DMA1_Stream6_IRQHandler         [WEAK]
                EXPORT  ADC_IRQHandler                  [WEAK]
                EXPORT  CAN1_TX_IRQHandler              [WEAK]
                EXPORT  CAN1_RX0_IRQHandler             [WEAK]
                EXPORT  CAN1_RX1_IRQHandler             [WEAK]
                EXPORT  CAN1_SCE_IRQHandler             [WEAK]
                EXPORT  EXTI9_5_IRQHandler              [WEAK]
                EXPORT  TIM1_BRK_TIM9_IRQHandler        [WEAK]
                EXPORT  TIM1_UP_TIM10_IRQHandler        [WEAK]
```

Gestión de las interrupciones

- La aplicación simplemente debe implementar la función manejador de la interrupción correspondiente:
  - stm32f4xx_it.c

```
/**
  ******************************************************************************
  * @file    /stm32f4xx_it.c
  * @author  MCD Application Team
  * @version V1.0.0
  * @date    19-September-2011
  * @brief   Main Interrupt Service Routines.
  *          This file provides template for all exceptions handler and
  *          peripherals interrupt service routine.
  ******************************************************************************
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_it.h"

/******************************************************************************/
/*                 STM32F4xx Peripherals Interrupt Handlers                   */
/*  Add here the Interrupt Handler for the used peripheral(s) (PPP), for the  */
/*  available peripheral interrupt handler's name please refer to the startup */
/*  file (startup_stm32f4xx.s).                                               */
/******************************************************************************/

/**
  * @brief  This function handles External line 0 interrupt request.
  * @param  None
  * @retval None
  */
void EXTI0_IRQHandler(void)
{
}

/**
  * @brief  This function handles TIM3 global interrupt request.
  * @param  None
  * @retval None
  */
void TIM3_IRQHandler(void)
{
}

/**
  * @
```

Gestión de las interrupciones

- core_cm4.h / .c
  - Contiene diversas funciones que proporcionan acceso al NVIC y permiten su configuración

Name	Core	Parameter	Description
void NVIC_SetPriorityGrouping (uint32_t PriorityGroup)	M3, M4	Priority Grouping Value	Set the Priority Grouping (Groups . Subgroups)
uint32_t NVIC_GetPriorityGrouping (void)	M3, M4	(void)	Get the Priority Grouping (Groups . Subgroups)
void NVIC_EnableIRQ (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Enable IRQn
void NVIC_DisableIRQ (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Disable IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Return 1 if IRQn is pending else 0
void NVIC_SetPendingIRQ (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Set IRQn Pending
void NVIC_ClearPendingIRQ (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Clear IRQn Pending Status
uint32_t NVIC_GetActive (IRQn_Type IRQn)	M3, M4	IRQ Number	Return 1 if IRQn is active else 0
void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)	M0, M3, M4	IRQ Number, Priority	Set Priority for IRQn (not threadsafe for Cortex-M0)
uint32_t NVIC_GetPriority (IRQn_Type IRQn)	M0, M3, M4	IRQ Number	Get Priority for IRQn
uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)	M3, M4	IRQ Number, Priority Group, Preemptive Priority, Sub Priority	Encode priority for given group, preemptive and sub priority
void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)	M3, M4	Priority, Priority Group, pointer to Preempt. Priority, pointer to Sub Priority	Decode given priority to group, preemptive and sub priority
void NVIC_SystemReset (void)	M0, M3, M4	(void)	Resets the System

Interrupciones externas

- stm32f4xx_exti.c

```
* @brief   This file provides firmware functions to manage the following
*          functionalities of the EXTI peripheral:
*           - Initialization and Configuration
*           - Interrupts and flags management
*
*  @verbatim
*
*          ===================================================================
*                                EXTI features
*          ===================================================================
*
*          External interrupt/event lines are mapped as following:
*            1- All available GPIO pins are connected to the 16 external
*               interrupt/event lines from EXTI0 to EXTI15.
*            2- EXTI line 16 is connected to the PVD Output
*            3- EXTI line 17 is connected to the RTC Alarm event
*            4- EXTI line 18 is connected to the USB OTG FS Wakeup from suspend event
*            5- EXTI line 19 is connected to the Ethernet Wakeup event
*            6- EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
*            7- EXTI line 21 is connected to the RTC Tamper and Time Stamp events
*            8- EXTI line 22 is connected to the RTC Wakeup event
*
*          ===================================================================
*                              How to use this driver
*          ===================================================================
*
*          In order to use an I/O pin as an external interrupt source, follow
*          steps below:
*            1- Configure the I/O in input mode using GPIO_Init()
*            2- Select the input source pin for the EXTI line using SYSCFG_EXTILineConfig()
*            3- Select the mode(interrupt, event) and configure the trigger
*               selection (Rising, falling or both) using EXTI_Init()
*            4- Configure NVIC IRQ channel mapped to the EXTI line using NVIC_Init()
```

Interrupciones externas

- stm32f4xx_exti.c

```
/* Initialization and Configuration functions *****************************/
void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct);

/* Interrupts and flags management functions ******************************/
ITStatus EXTI_GetITStatus(uint32_t EXTI_Line);
void EXTI_ClearITPendingBit(uint32_t EXTI_Line);

...
```

Ejemplo

- Este ejemplo muestra cómo configurar y manejar interrupciones externas:
  - La línea *EXTI Line0* (conectada al pin *PA0 → Pulsador azul*) se ha configurado para generar una interrupción en cada flanco de subida. En la correspondiente rutina de interrupción el LED conectado al pin PD.12 va cambiando de estado.
- main.c

```
48  int main(void)
49  {
50    /*!< At this stage the microcontroller clock setting is already configured,
51         this is done through SystemInit() function which is called from startup
52         file (startup_stm32f4xx.s) before to branch to application main.
53         To reconfigure the default setting of SystemInit() function, refer to
54          system_stm32f4xx.c file
55       */
56
57    /* Initialize LEDs mounted on STM32F4-Discovery board */
58    STM_EVAL_LEDInit(LED3);
59
60    /* Configure EXTI Line0 (connected to PA0 pin) in interrupt mode */
61    EXTILine0_Config();
62
63    /* Generate software interrupt: simulate a rising edge applied on EXTI0 line */
64    //EXTI_GenerateSWInterrupt(EXTI_Line0);
65
66    while (1)
67    {
68    }
69  }
```

Ejemplo

- main.c

```
void EXTILine0_Config(void)
{

  GPIO_InitTypeDef   GPIO_InitStructure;
  NVIC_InitTypeDef   NVIC_InitStructure;

  /* Enable GPIOA clock */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  /* Enable SYSCFG clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

  /* Configure PA0 pin as input floating */
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* Connect EXTI Line0 to PA0 pin */
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);

  /* Configure EXTI Line0 */
  EXTI_InitStructure.EXTI_Line = EXTI_Line0;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

  /* Enable and set EXTI Line0 Interrupt to the lowest priority */
  NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

Ejemplo

- stm32f4xx_it.c

```
/**
  * @brief  This function handles External line 0 interrupt request.
  * @param  None
  * @retval None
  */
void EXTI0_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line0) != RESET)
  {
    /* Toggle LED4 */
    STM_EVAL_LEDToggle(LED3);

    /* Clear the EXTI line 0 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line0);
  }
}
```

Ejercicios

- Ahora tu ... prueba el ejemplo y analízalo
- Modifica el ejemplo para contabilizar y mostrar el número de veces que se ha producido la interrupción..

Interrupciones

- Trabajo:
  - Intentad realizar la gestión de la visualización de un display como se muestra en la figura en base al uso interrupciones de modo que cada paso de giro del potenciómetro -simulado mediante pulsaciones del botón de usario- genere la interrupción externa 0 que incrementará/ decrementará la cuenta. Se supone que es una rueda que gira en un solo sentido y que cuando llega al tope -99- vuelve a cero).

¡¡¡Ayuda!!!

  - Manual de la “STM32F4 DSP and standard peripherals library”
  - http://armcortexm.blogs.upv.es/material-del-curso/