Sistemas Embebidos

https://blog.granted.com/2014/11/16/2-tips-for-avoiding-interruptions-at-work/

Interrupciones … un mal necesario

Ángel Perles

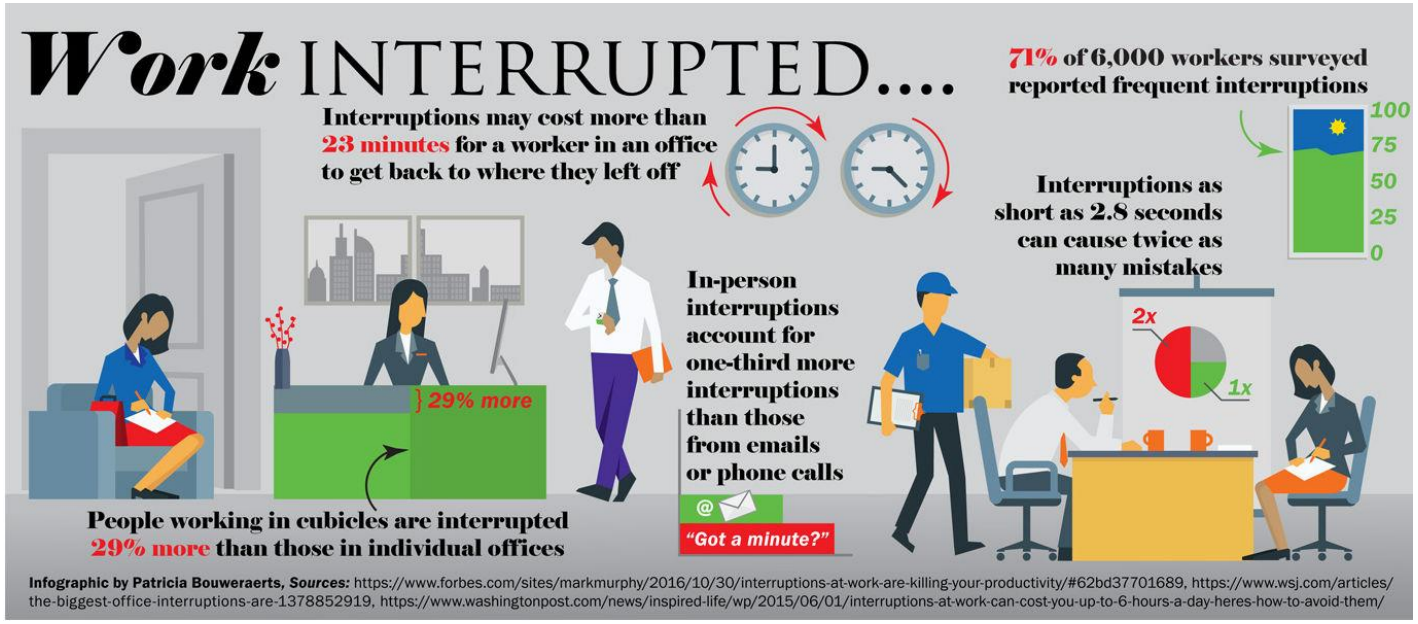
r06

Contenido

- Objetivos
- Interrupciones estilo “HAL”
- Prioridades (a quién le toca)
- Secciones críticas
- Manejadores razonables
- Con el cube
- (Cajón de sastre)

Objetivos

- Comprender las implicaciones del mecanismo de interrupciones
- Gestionar la confluencia de interrupciones (prioridades)
- Reducir los efectos “malignos” de las interrupciones



Interrupciones “HAL”

- Las “HAL” ¿simplifican? los manejadores de interrupción …
  - Proporcionando funciones para gestionar los *flags*
  - Proporcionando una función “*callback*”
- Para entenderlo, veamos un manejador EXTI**x** estilo HAL

```
void EXTI0_IRQHandler(void)
{
  // St’s HAL proposes to call this predefined function
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

- **NOTA:** ¿Ves la “x” roja de arriba?

Interrupciones “HAL”

- Las EXTx disponibles en el STM32L476
  - **EXTI0**
  - **EXTI1**
  - **EXTI2**
  - **EXTI3**
  - **EXTI4**
  - **EXTI9_5**
  - **EXTI15_10**

Interrupciones “HAL”

- Veamos qué ha escrito St en esa función

```
/**
  * @brief  This function handles EXTI interrupt request.
  * @param  GPIO_Pin: Specifies the pins connected EXTI line
  * @retval None
  */
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);
  }
}

/**
  * @brief  EXTI line detection callbacks.
  * @param  GPIO_Pin: Specifies the pins connected EXTI line
  * @retval None
  */
__weak void HAL_GPIO_EXTI_Callback(uint16_t 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
   */
}
```

- ¿Lo reconoces?. Fíjate también en el “*weak*”

Interrupciones “HAL”

- Y un ejemplo de un posible manejador nuestro
  - Es una **reimplementación** del “*callback*”

```
#include <led.h>

// ...

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  static uint8_t estado = 0;
  if(GPIO_Pin == GPIO_PIN_0)
  {
    if (estado == 0) {
       estado = 1;
       LED_On();
    } else {
       estado = 0;
       LED_Off();
    }
  }
}
```

Prioridades (a quién le toca)

DISCA
DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

- El NVIC (Nested Vectored Interrupt Controller) decide
  - Preemptivo: interrupción de mayor prioridad interrumpe a las de menor
  - Hasta 256 niveles (números) de interrupción
  - Número menor -> mayor prioridad

```
/**
 * @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 
*/
  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
```

Prioridades (a quién le toca)

- Con tantísimo cacharro, es fácil que lleguen varias “simultáneamente”
- Hace falta un mecanismo para flexibilizar el orden
- El NVIC permite reclasificar interrupciones en grupos y subgrupos.
- Accesible fácilmente a través del CMSIS o del HAL

```
void HAL_NVIC_SetPriorityGrouping ( uint32_t PriorityGroup)
```

Sets the priority grouping field (pre-emption priority and subpriority) using the required unlock sequence.

```
void HAL_NVIC_SetPriority ( IRQn_Type IRQn, uint32_t PreemptPriority,
    uint32_t SubPriority)
```

Sets the priority of an interrupt.

Prioridades (a quién le toca)

DISCA
DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

- Localiza esto en el manual que toca ¿por qué está ahí?

HAL_NVIC_SetPriorityGrouping

Function name	**void HAL_NVIC_SetPriorityGrouping (uint32_t PriorityGroup)**
Function description	Set the priority grouping field (pre-emption priority and subpriority) using the required unlock sequence.
Parameters	• **PriorityGroup:** The priority grouping bits length. This parameter can be one of the following values: – NVIC_PRIORITYGROUP_0: 0 bit for pre-emption priority, 4 bits for subpriority – NVIC_PRIORITYGROUP_1: 1 bit for pre-emption priority, 3 bits for subpriority – NVIC_PRIORITYGROUP_2: 2 bits for pre-emption priority, 2 bits for subpriority – NVIC_PRIORITYGROUP_3: 3 bits for pre-emption priority, 1 bit for subpriority – NVIC_PRIORITYGROUP_4: 4 bits for pre-emption priority, 0 bit for subpriority
Return values	• **None:**
Notes	• When the NVIC_PriorityGroup_0 is selected, IRQ pre-emption is no more possible. The pending IRQ priority will be managed only by the subpriority.

Prioridades (a quién le toca)

- Y esto

HAL_NVIC_SetPriority

Function name	**void HAL_NVIC_SetPriority (IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)**
Function description	Set the priority of an interrupt.
Parameters	• **IRQn:** External interrupt number. This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l4xxxx.h)) • **PreemptPriority:** The pre-emption priority for the IRQn channel. This parameter can be a value between 0 and 15 A lower priority value indicates a higher priority • **SubPriority:** the subpriority level for the IRQ channel. This parameter can be a value between 0 and 15 A lower priority value indicates a higher priority.
Return values	• **None:**

Prioridades (a quien le toca)

DISCA
DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

- Básicamente
  - Una interrupción con una “preempt priority” mayor (número menor) interrumpe una de “preempt priority” menor
  - Si dos con al misma “preempt priority” llegan simultánemamente, se atenderá primero a la de “subpriority” más alta.

- Es bástannnnte confuso. Lectura para mañana si estáis fresquitos

Cutting Through the Confusion with Arm Cortex-M Interrupt Priorities
https://community.arm.com/iot/embedded/b/embedded-blog/posts/cutting-through-the-confusion-with-arm-cortex-m-interrupt-priorities

Prioridades (a quien le toca)

DISCA
DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

- Ejemplo
  - Al principio de la aplicación, ponemos el “grouping” ese. Por ejemplo,

```
HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUPING_4);
```

- Y usamos la otra función para cambiar el comportamiento “normal”
  - Ejemplo, SYSTICK deja de ser la más importante

```
HAL_NVIC_SetPriority(SYSTICK_IRQn,15,0);
```

- Si echamos un vistazo a la implementación de HAL_Init() podemos comprobar qué configuración ha decidido usar St
  - Actualmente *NVIC_PRIORITYGROUPING_4*

Secciones críticas

- Contamos tantas mentiras en las asignaturas …
- El mecanismo de interrupciones provoca que las aplicaciones se “rompan” en cualquier punto de ejecución del código máquina
- Este efecto está presente también en los “microkernels” e incluso en tu Android o tu Windows
- Algo que no se debe romper para no provocar una inconsistencia se llama “sección crítica”
- Solución “cutre” de este año

```
__enable_irq();
// code here Get out as fast as possible!!!!!
__disable_irq();
```

- Muy avanzado. Pero sabed que es un problema si trabajáis en esto

Manejadores razonables

- Piensa que en las HAL, la interrupción SYSTICK se produce <u>1000 veces en un segundo</u>

```c
/**
* @brief This function handles System tick timer.
*/
void SysTick_Handler(void)
{
  HAL_IncTick();
  HAL_SYSTICK_IRQHandler();
}
```

- Pues como pongas cuatro cosas más mal hechas, hundes al micro
- El manejador debe ser
  - Cortito
  - Rápido
  - No esperar a nadie
  - Y mejor si no existiese. Una interrupción es lo peor de los sistemas de tiempo real … pero va a ser que no

Manejadores razonables

- Una gran estrategia es dividir la tarea en lo que hace la interrupción y lo que hace el bucle principal

```
extern volatile uint8_t _button_state;

void EXTI0_IRQHandler(void)
{
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_PIN_0) != RESET) {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_0);

    _button_state = 1;  // button pressed

  }
}
```

- ¡Uy! No es estilo HAL. Ja, ja, para la práctica.

Manejadores razonables

- Una gran estrategia

```c
#include <stdio.h>
#include "stm32f4xx_hal.h"
#include <button.h>

volatile uint8_t _button_state = 0;

uint8_t button_GetPressed(void)
{
   return _button_state;
}

void button_Reset(void)
{
   _button_state = 0;   // not pressed
}

int main(void)
{
   // ...

   for(;;)
   {
      // check if the button was pressed in the past
      if (button_GetPressed()) {
         printf("Se ha pulsado el botoncito\n");
         button_Reset();
      }

      // simulate other tasks
      for (i=0;i<100000000;i++) {};
      printf("Tarari!\n");

   }; //forever i que no siga res!

}
```

Con el cube

- En directo … si me acuerdo
  - Generar un interrupción externa del pulsador azul de la Nucleo-64 STM32L476RG

(Cajón de sastre)

- Añadir inlinning
- Añadir assembler incrustado en el código

The End