Interrupciones

Interrupciones.
Concepto.

- Interrupción: Es una señal que pide al procesador suspender temporalmente la ejecución del programa para ejecutar una rutina de atención a la interrupción.

- Ejemplo.

Interrupciones.
Ventajas.

- Se puede lograr un tiempo de respuesta muy corto para las peticiones.
- Las operaciones urgentes se pueden programar independientemente del código principal.
- La interrupción se puede desencadenar por una señal externa:
  - IRQ petición de interrupción, desde pines dedicados.
  - Cambio de valor lógico en pines digitales.
- La interrupción se puede desencadenar por mecanismos internos:
  - Finalización de un contador
  - Excepción en una instrucción o operación aritmética
  - Petición mediante interrupción software
  - Otros (Controlador de alimentación, etc)

Interrupciones.
Mecanismo de Interrupción.

- Una vez se recibe y acepta una petición de interrupción, el procesador realiza las siguientes operaciones:
  - Finaliza la ejecución de la instrucción actual (salvo con alguna excepción como el RESET)
  - Se almacena un puntero a la siguiente instrucción a ejecutar en la pila.
  - Se guarda **el contexto** (registros, flags…)
  - La dirección de la rutina de atención a la interrupción se lee del adecuado vector de interrupción (de acuerdo a la fuente que ha pedido la interrupción).
  - Se ejecuta la rutina de atención a la interrupción.
  - Al finalizar la rutina de atención a la interrupción, el procesador recupera de la pila la dirección de la siguiente instrucción a ejecutar del programa principal.

Interrupciones.
Control de las Interrupciones.

- Algunas operaciones son críticas y nunca pueden interrumpirse. Es necesario deshabilitar las interrupciones antes y volverlas a habilitar después de su ejecución.
- Algunos procesadores permiten asignar prioridades a las interrupciones de diferentes fuentes. Esto permite resolver cuando se producen varias interrupciones simultáneas.
- Habilitar y Deshabilitar interrupciones se suele hacer mediante instrucciones específicas o mediante registros dedicados.
- NOTAS:
  - En el arranque las interrupciones están deshabilitadas por defecto, para que el programa arranque correctamente.
  - Cuando se dispara una interrupción algunos procesadores deshabilitan automáticamente las interrupciones menores de la actual.
  - Cuando se recibe una petición de interrupción el procesador activa un FLAG indicando la petición de interrupción. Este flag debe ser limpiado (puesto a cero) por la rutina de atención a la interrupción.
  - Algunos procesadores tienen una fuente o nivel de interrupción que no se puede deshabilitar (NMI, Non Maskable Interrupt). Su uso se limita a situaciones excepcionales.

Interrupciones.
Control de las Interrupciones.

- El correcto funcionamiento de un programa no se debe ver afectado por las interrupciones.
- Es obligatorio que las rutinas de atención a las interrupciones dejen el estado del procesador sin modificar (registros, flags, etc.).
- Esto se consigue guardando el entorno (contexto) al principio de las rutinas de atención a la interrupción y restaurándolo al final.
- NOTAS:
  - El entorno (contexto) se guarda en la pila o en un área de memoria específica.
  - Algunos procesadores guardan automáticamente el entorno (total o parcialmente) cuando se atiende a la interrupción, otros no.
  - Muchas veces tienen instrucciones específicas para guardar el entorno.
  - Los procesadores que automáticamente deshabilitan las interrupciones cuando saltan a la rutina de atención también las habilitan automáticamente.

Interrupciones.
Programar con Instrucciones de Arduino

- Los compiladores tienen mecanismos para programar interrupciones sin necesidad de hacerlo en lenguaje ensamblador.
  - Algunas funciones se pueden diseñar como rutinas de atención a la interrupción con la palabra interrupt.
  - Algunos compiladores automáticamente insertan instrucciones para guardar y recuperar el entorno, además de configurar el vector de interrupción.
  - Habilitar y Deshabilitar interrupciones se suele hacer con macros o directivas de compilación (en Arduino se utilizan las funciones interrupts() y noInterrupts() respectivamente).
  - Para informar que el valor de una variable se modifica en una rutina de atención de interrupción se utiliza volatile.
  - Las peticiones de interrupción son impredecibles, eso complica el intercambio de datos entre el programa principal y la rutina de atención a la interrupción.

Interrupciones. Ejemplo. Mala solución.

- Controlador de temperatura: La alarma debe sonar si dos medidas de temperatura realizadas con la rutina de interrupción son diferentes.
- Haciendo la comparación entre las dos medidas en una sóla instrucción en C no soluciona el problema. (Esa instrucción se compila con varias instrucciones en código máquina).
- Este tipo de fallos es difícil de detectar y de reproducir.

```
static volatile int temp[2];

interrupt void measure(void)
{
  temp[0] = !! first measurement;
  temp[1] = !! second measurement;
}

void controller(void)
{
  int temp0, temp1;
  for (;;)
    {
      temp0 = temp[0];
      temp1 = temp[1];
      if (temp0 != temp1) !! sound the alarm;
    }
}
```

```
...
void controller(void)
{
  for (;;)
    if (temp[0] != temp[1]) !! sound the alarm;
}
...
```

Interrupciones.
Ejemplo. Solución correcta

- Las instrucciones que leen la medida de temperatura envían mediante la rutina de la interrupción al controlador forman una sección crítica que no se puede interrumpir .
- En Arduino utilizamos las funciones:
  - noInterrupts(); //deshabilita
  - Código a ejecutar en un bloque.
  - Interrupts(); //habilita

```
static volatile int temp[2];

interrupt void measure(void)
{
  temp[0] = !! first measurement;
  temp[1] = !! second measurement;
}

void controller(void)
{
  int temp0, temp1;
  for (;;)
    {
      disable();   /* Disable interrupts */
      temp0 = temp[0];
      temp1 = temp[1];
      enable();    /* Reenable interrupts */

      if (temp0 != temp1) !! sound the alarm;
    }
}
```

Interrupciones. Ejemplo. Otra Solución.

- Esta solución no requiere deshabilitar interrupciones.
- El programa principal a veces tiene que realizar una iteración extra inútil antes de activar la alarma.

```
static volatile int temp_a[2], temp_b[2];
static int controller_uses_b = 0;

interrupt void measure(void)
{
  if (controller_uses_b)
    {
      temp_a[0] = !! first measurement;
      temp_a[1] = !! second measurement;
    }
  else
    {
      temp_b[0] = !! first measurement;
      temp_b[1] = !! second measurement;
    }
}

void controller(void)
{
  for (;; controller_uses_b = !controller_uses_b)
    if (controller_uses_b)
      {
        if (temp_b[0] != temp_b[1]) !! sound the alarm;
      }
    else
      if (temp_a[0] != temp_a[1]) !! sound the alarm;
}
```

Interrupciones.
Ejemplo. Solución Mejorada.

- Para que esta solución sea correcta es necesario que la instrucción last+=2; se ejecute en un sólo bloque.
- En la práctica deshabilitar interrupciones durante las comunicaciones con rutinas de interrupción es aceptable en la mayoría de los casos.
- La solución más compleja se usa sólo cuando deshabilitar las interrupciones es imposible o está prohibido.

```
#define MAX_FIFO 10     /* Must be even ! */
static volatile int temp_fifo[MAX_FIFO];
static volatile int first = 0;
static int last = 0;

interrupt void measure(void)
{
  /* If the buffer is not saturated */
  if (!((first + 2 == last)
      || (first == MAX_FIFO - 2 && last == 0)))
    {
      temp_fifo[first] = !! first measurement;
      temp_fifo[first + 1] = !! second measurement;
      first += 2;
      if (first == MAX_FIFO)
        first = 0;
    }
  else   !! discard measurements;
}

void controller(void)
{
  int temp0, temp1;

  for (;;)
    if (first != last)    /* If the buffer is not empty */
      {
        temp0 = temp_fifo[last];
        temp1 = temp_fifo[last + 1];
        last += 2;
        if (last == MAX_FIFO)
          last = 0;
        if (temp0 != temp1) !! sound the alarm;
      }
}
```

Interrupciones.
Programar con Esp-idf

- Esp-idf es un framework de programación desarrollado por Expressif para controlar a bajo nivel el ESP32
- Página web: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
- Interrupciones:
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/hlinterrupts.html
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/intr_alloc.html
- Configurar pines para realizar interrupciones externas:
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html

Configurar pines

Instalar la rutina de atención a la interrupción

Asignar el manejador de la interrupción al pin correspondiente

Interrupciones.
Programar con Esp-idf

Configurar pines

struct **gpio_config_t**

Configuration parameters of GPIO pad for gpio_config function.

Public Members

uint64_t **pin_bit_mask**

GPIO pin: set with bit mask, each bit maps to a GPIO

Máscara de pines: pone a 1 los bits que corresponden con los GPIO que queremos configurar

gpio_mode_t **mode**

GPIO mode: set input/output mode

GPIO_MODE_DISABLE / GPIO_MODE_INPUT/GPIO_MODE_OUTPUT/ GPIO_MODE_OUTPUT_OD / GPIO_MODE_INPUT_OUTPUT_OD / GPIO_MODE_INPUT_OUTPUT

gpio_pullup_t **pull_up_en**

GPIO pull-up

GPIO_PULLUP_DISABLE / GPIO_PULLUP_ENABLE

gpio_pulldown_t **pull_down_en**

GPIO pull-down

GPIO_PULLDOWN_DISABLE / GPIO_PULLDOWN_ENABLE

gpio_int_type_t **intr_type**

GPIO interrupt type

GPIO_INTR_DISABLE / GPIO_INTR_POSEDGE / GPIO_INTR_NEGEDGE/ GPIO_INTR_ANYEDGE / GPIO_INTR_LOW_LEVEL / GPIO_INTR_HIGH_LEVEL

Interrupciones.
Programar con Esp-idf

Instalar la rutina de atención a la interrupción

```
esp_err_t gpio_install_isr_service(int intr_alloc_flags)
```

Parameters: **intr_alloc_flags** – Flags used to allocate the interrupt. One or multiple (ORred) ESP_INTR_FLAG_* values. See esp_intr_alloc.h for more info.

Returns:

- ESP_OK Success
- ESP_ERR_NO_MEM No memory to install this service
- ESP_ERR_INVALID_STATE ISR service already installed.
- ESP_ERR_NOT_FOUND No free interrupt found with the specified flags
- ESP_ERR_INVALID_ARG GPIO error

ESP_INTR_FLAG_IRAM: se instala en la memoria RAM interna
ESP_INTR_FLAG_LEVELX: Nivel X de interrupción (1-7)

Interrupciones.
Programar con Esp-idf

Asignar el manejador de la interrupción al pin correspondiente

esp_err_t **gpio_isr_handler_add**(gpio_num_t gpio_num, gpio_isr_t isr_handler, void *args)

Parameters:

- **gpio_num** – GPIO number
- **isr_handler** – ISR handler function for the corresponding GPIO number.
- **args** – parameter for ISR handler.

Returns:

- ESP_OK Success
- ESP_ERR_INVALID_STATE Wrong state, the ISR service has not been initialized.
- ESP_ERR_INVALID_ARG Parameter error

GPIO_NUM_x

Interrupciones.
Latencia.

- El retardo entre una petición de interrupción I y el final de la ejecución de las operaciones urgentes en la rutina de atención de la interrupción R_I se llama tiempo de respuesta o latencia de la interrupción.
- Esta latencia depende de cuatro parámetros:
  1. El intervalo más largo durante el cual están deshabilitadas las interrupciones de prioridad mayor o igual a I
  2. El tiempo necesario para ejecutar las rutinas de interrupciones con prioridad mayor que la interrupción R_I .
  3. El retardo máximo entre la petición de interrupción y el salto a la rutina de atención a la interrupción.
  4. El tiempo empleado en R_I antes de haber ejecutado las operaciones urgentes.
- Una buena estrategia es:
  - Deshabilitar las interrupciones por el mínimo tiempo posible (1).
  - Realizar las rutinas de atención a las interrupciones rápida y eficientemente (2 y 4).
  - (3) Es una característica del procesador y no puede influir el programador.

Interrupciones. Ejemplo 1.

- Modificar el programa que lee el estado de un pin GPIO5 para que cuando cambie el valor del pin parpadee la mitad inferior de la pantalla 5 veces en color rojo.
- Modificar el valor del evento y observar lo que sucede en cada caso.
- Se asignará una interrupción a ese pin GPIO mediante la instrucción de Arduino:
  - attachInterrupt(digitalPinToInterrupt("Nº de pin"), "función de interrupción", "evento");
  - Nº de pin: pin que se quiere testear
  - Función de interrupción: Rutina de atención a la interrupción.
  - Evento: Seleccionar entre LOW, CHANGE, FALLING y RISING.

EjemploGTI_03_Interrupt

```
#include <M5Stack.h>

#define PinGPIO G5
static volatile uint8_t active=0;

void setup() {
  // put your setup code here, to run once:
  //Parametros de begin(bool LCDEnable, bool SDEnable, bool SerialEnable)
M5.begin(true,false,true);
M5.Lcd.setTextSize(2);
M5.Lcd.printf("Pin %d  value:  ",PinGPIO);
pinMode(PinGPIO,INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(PinGPIO), blinka, RISING);
  //LOW,CHANGE,FALLING,RISING
}

void blinka()
{
    M5.Lcd.print ("        ROJO");
    active=10;
}

void loop() {
  // put your main code here, to run repeatedly:
  M5.Lcd.setCursor(0, 16);
  if (digitalRead(PinGPIO)){M5.Lcd.print ("ON ");}
  else {M5.Lcd.print ("OFF");};
  if (active)
  {
    if (active%2) M5.Lcd.fillRect(0, 100, 300, 150, RED);
    else M5.Lcd.fillRect(0, 100, 300, 150, BLACK);
    active--;
  }
  M5.update();
  delay(500);
}
```

Interrupciones.
Ejemplo 2.

- Realizar una aplicación utilizando el framework Esp-idf,
- Configure el pin GPIO5 para que se produzca una interrupción cuando lo conectemos a nivel bajo
- La rutina de atención a la interrupción deberá sacar por la pantalla LCD el número de veces que se ha producido la interrupción
- Amplie el programa con una interrupción asociada al GPIO2 que se produzca cuando lo conectemos a nivel alto. La interrupción deberá descontar una unidad del contador anterior
- Asígneles niveles de interrupción diferentes, y compruebe que el sistema funciona correctamente

Bibliografía del tema

- An Embedded Software Primer (David E. Simon) Capítulo 4
- esp-idf framework:
  - Interrupciones de alto nivel https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/hlinterrupts.html
  - Asignación de interrupciones: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/intr_alloc.html
  - Configuración GPIO: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html