

**INGENIERIA EN MECATRONICA**

**Asignatura: SISTEMAS EMBEBIDOS**

**Maestro: Carlos Enrique Moran Garabito**

**Alumno: Mario Alcalá Villagómez 8A**

**Matricula: 17310857**

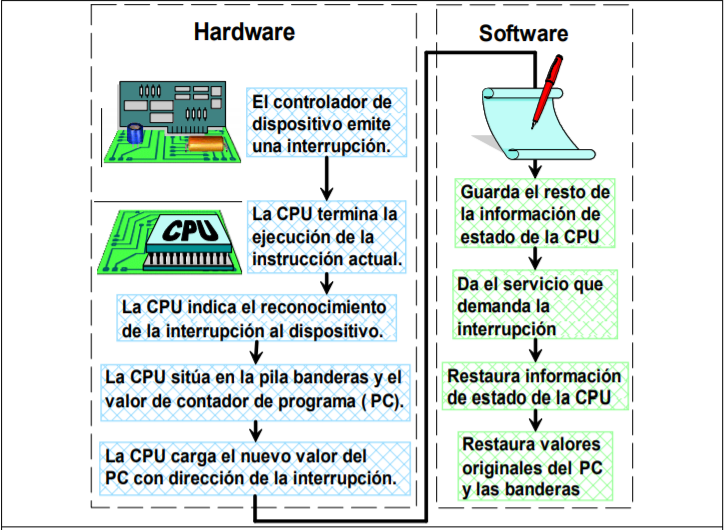
**Periodo: Enero - Abril 20**

**Lugar:** [Universidad Politécnica de la Zona Metropolitana de Guadalajara](https://upzmg.edu.jalisco.gob.mx/lugar/13061)

Carretera Tlajomulco-Santa Fe, km 3.5 #595, Colonia: Lomas de Tejeda, CP: 45670   
Municipio: Tlajomulco de Zúñiga

**Los interruptores.**

Las interrupciones son cambios en el flujo de control, no ocasionados por el programa que se ejecuta, sino por algún otro suceso que necesita el servicio inmediato de la CPU por lo general relacionado con los dispositivos de E/S. Por ejemplo, un programa puede pedirle al controlador de disco que empiece a transferir información y que genere una interrupción cuando acabe la transferencia.

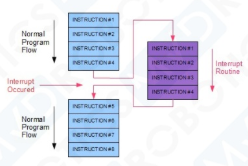


La Figura muestra, en forma muy simplificada, los pasos y los componentes involucrados en el manejo de una interrupción. La señal de petición de interrupción provoca que la CPU detenga el programa en curso, salve su estado (es decir, se guardan todos los contenidos de los registros de la CPU) y transfiera el control a una Rutina de Servicio de Interrupción, o ISR (del inglés “Interrupt Service Routine”) la cual realiza alguna acción apropiada para darle servicio a la petición. Al terminar el servicio de la interrupción, se debe continuar el código interrumpido exactamente en él mismo estado en que estaba cuando tuvo lugar la interrupción, lo cual se logra restaurando los registros internos al estado que tenían antes de la interrupción previamente salvado permitiendo continuar el flujo normal de procesamiento.

En los sistemas embebidos el tiempo es un elemento muy importante.

Un sistema embebido tiene, generalmente, que atender varias “tareas”, algunas de ellas periódicas, otras disparadas por eventos.

Las interrupciones y los timers son herramientas clave para lograr un manejo efectivo del tiempo y los eventos en un sistema embebido.



En la programación estándar, en una PC, las interrupciones son un recurso del hardware que usa a nivel del S.O. (Sistema Operativo), el kernel, los drivers, lo que llamamos “bajo nivel”. En un microcontrolador la programación es siempre de “bajo nivel”. Programamos directamente sobre el CPU, y no sobre la “máquina virtual” de S.O.

Asimismo, la programación normal de la PC, bajo el S.O., existen los threads. El equivalente en el microcontrolador son las interrupciones. Aparecen algunos problemas parecidos, de sincronización, y se resuelven de manera similar.

Las interrupciones son un recurso esencial de los sistemas embebidos. Básicamente, la interrupción es un mecanismo mediante el cual el CPU puede, ante cierto evento, suspender lo que está haciendo en ese momento y pasar a atender una rutina de alta prioridad. Una vez finalizada esta, el CPU vuelve a su actividad anterior.

La interrupción es disparada por un evento externo al CPU. Puede ser el cambio de estado de un pin (interrupción externa) o cierta señal de un dispositivo interno del microcontrolador, por ejemplo, el desborde de un timer, el estado del ADC, etc.

En algunos procesadores también existen las interrupciones generadas por software (lo cual tiene sentido en el contexto de un S.O.). En el AVR no existen, pero se pueden simular cambiando manualmente el estado de un pin.

El timbre de una casa, o el “ring” del teléfono, son analogías perfectas de lo que es y cómo funciona una interrupción.

Cuando el CPU recibe la señal de interrupción, abandona (interrumpe) inmediatamente lo que está haciendo y salta a una rutina especial. Dicha rutina suele llamarse Interrupt Handler o Interrupt Service (ISR).

Una vez que la ISR termina, el CPU retoma su actividad anterior desde el punto en que la dejo.

Antes de atender la interrupción, el CPU guarda el estado de sus registros (incluyendo el Program Counter), para poder continuar luego con la ejecución de las instrucciones del programa principal. Ese “estado” del CPU se llama el contexto, la operación de salvarlo y restituirlo se llama cambio de contexto.

Las interrupciones son señales aleatorias, externas al CPU, que pueden ocurrir en cualquier momento, fuera del flujo predecible de un programa. Es un proceso asíncrono; funciona como si fuera otro hilo de ejecución (thread).

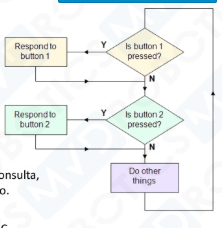
El microcontrolador dispone de mecanismos para activar, desactivar, priorizar, inhibir, etc., distintas fuentes de interrupciones, y para cada una de ellas con rutinas ISR programadas por el usuario.

**Polling vs Interrupciones.**

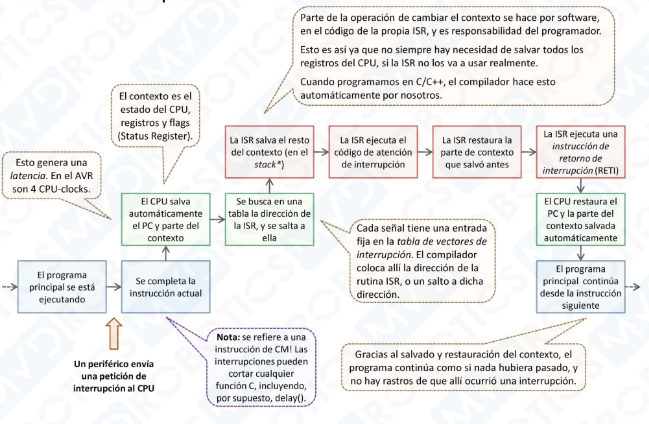
Las interrupciones son lo que permite evitar el polling.

El polling consiste en consultar permanentemente la ocurrencia de cierto evento desde el programa

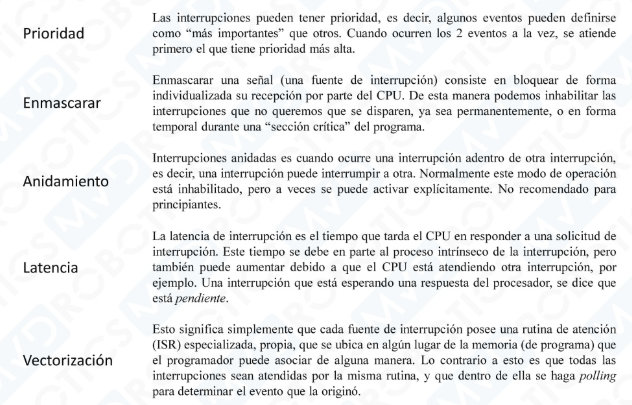
Al contrario que la interrupción, el polling es sincronico y determinista.



**Proceso de interrupción.**



**Tópicos y terminología de las interrupciones**.



**Fuentes de interrupción**

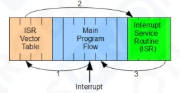
En el AVR, diversas condiciones pueden generar una interrupción, por ejemplo:

Obsérvese que a veces se tiene el mismo vector para señalizar varias condiciones distintas, en cuyo caso, la ISR debe “investigar” qué fue lo que pasó.

Tabla de vectores de interrupción

Cada evento tiene asociada una dirección de memoria en donde se aloja el código especifico a ejecutar, en la llamada tabla de vectores de interrupción, que se ubica al comienzo de la memoria programada.

En esa dirección, el copilado coloca un “jmp” a la dirección de la ISR, que pue estar en cualquier lado.



Esta tabla también establece la prioridad de cada interrupción, que en el caso del AVR, es fija. Cuando más bajo el numero en la tabla, más alta la prioridad del evento.

