Antes de abordar lo que es una interrupción como tal, expondremos la motivación que llevo a implementarlas dentro de los microporcesadores, para poder utilizarlos para afrontar principalmente problemas que requieren de la interacción con el mundo real.

Imagine que hoy llegará a su casa un envío importante, sin embargo el timbre de la puerta está averiado y usted no puede avisarle al repartidor sobre éste inconveniente, la primera opción que puede considerar consiste en esperar cerca de la puerta hasta que el envío llegue, pero ésta no es una muy buena idea pues quizá tenga muchas cosas que hacer, o simplemente no le parezca que estar sentado esperando sea una actividad muy enriquecedora, por lo cual usted opta por continuar con sus actividades regulares pero ir a revisar la puerta constantemente para ver si el repartir ha llegado con su envío; no obstante, ésta idea tampoco resulta ser muy prometedora puesto que el repartidor no puede quedarse esperando en la puerta hasta que usted aparezca y la probabilidad de que coincidan es ciertamente baja, además, a la larga perdería el tiempo que le tomaría ir a revisar la puerta en las veces que no encontró al repartidor.

Este problema, que dentro del ámbito de los microprocesadores consiste básicamente en ser capaz de detectar y responder a una señal que puede ocurrir en cualquier momento, y que no da tiempo a ser aplazada, se presenta en múltiples escenarios, ya sea dentro de un microcontrolador para un sistema embebido o incluso dentro de las actividades que se realizan comúnmente en un PC, por lo cual es un problema que resultar ser recurrente y de importante solución.

Dada la necesidad de poder atender estas señales en cualquier instante, una alternativa, justo como en la analogía del principio, pudo haber sido destinar al microprocesador a estar al pendiente exclusivamente de susodicha señal, lo cual evidentemente es un desperdicio total de la capacidad de procesamiento de este, que además imposibilita el reconocimiento de varias señales de este tipo; una segunda aproximación podría ser programar al microprocesador para que revisara periódicamente todos los periféricos verificando si alguno de ellos estaba emitido alguna señal, este método es conocido como *polling*, el cual fue el primero en ser implementado en respuesta a este problema, sin embargo, no era una solución muy acertada pues consumía recursos y tiempo al microprocesador revisar todos los periféricos, que a la larga lo hacían un proceso muy ineficiente, además de que señales que sólo se activasen durante periodos de tiempo inferiores al de revisión podrían llegar a ser omitidas, lo cual es incluso peor que la primera alternativa, por lo tanto se continuo con la búsqueda de una solución alterna, la cual concluyó con lo que se conoce dentro del ámbito de los microprocesadores como una interrupción.

Una interrupción no es más que un mecanismo cuyo propósito es poder hacer que el microprocesador detenga lo que está procesando, pase a ejecutar unas instrucciones previamente establecidas, las cuales se conocen técnicamente como una ISR (Interrupt Service Routines [Implementación UNO]), en el momento en que una señal prioritaria es recibida desde el exterior, como un periférico (el teclado o el mouse por ejemplo), o desde el interior, como en el caso del desbordamiento de una variable o una división por 0 [Manual], con el fin de atender esta señal y luego poder retomar el proceso anterior en las mismas condiciones en que las que fue interrumpido.

Hay varias fuentes de interrupción, dependen del PIC. Hay que habilitarlas para poder utilizarlas. Se revisa la bandera de interrupción para no tener que revisar todas las señales. La bandera se coloca en alto cuando se detecta una interrupción, incluso si dejo de ser detectada para que no sea omitida por los periodos de duración de la activación. Si se detecta una interrupción se desactiva una bandera para no recibir más interrupciones. Se termina de ejecutar la línea en curso y se procede a revisar si hay alguna interrupción. El vector de interrupción está en la posición 0004h. Las fuentes de interrupciones varían según el modelo del PIC. Interrupciones externas e internas. Las posiciones de las instrucciones que se deben ejecutar luego de una interrupción están en la tabla de vectorización. Hay registros para poder habilitar cada interrupción. Luego de activar una interrupción, según la tabla de vectorización se ejecuta una instrucción, que normalmente es de JMP (jump) para ir hacia otra zona de la memoria donde están todas las instrucciones correspondientes a la interrupción. Los microcontroladores de gama baja y media solo procesan interrupciones, pero los de gama alta procesan dos prioridades, alta y baja. No se produce borrado por hardware en las interrupciones del puerto serie y del timer 2. Registro IP (Interrupt Priority). Una interrupción de nivel bajo puede ser interrumpida por una de alto, pero no por otra de nivel bajo. Terminar ejecución RETI. Las interrupciones son revisadas en el estado 5 fase 2 de cada ciclo de máquina. Si está habilitada la individual y la global se termina la instrucción, se guarda la posición actual del programa principal, se ejecuta la subrutina y luego mediante RETI se regresa, algunas veces hay que rehabilitar la interrupción individual (MÁS INFO EN EL DOCUMENTO CON EJERCICIOS). Los micros traen incorporado un dispositivo que les permite controlar las interrupciones. Las tarjetas de Arduino incluyen una unidad de interrupciones. La rutina de interrupción se llama ISR. La tabla de vectores inicia con la interrupción RESET. La siguiente en prioridad es INT0 y luego INT1. El primer tipo de interrupción es por nivel de voltaje (lógico) y flancos de subida o de bajada y se limpian por hardware. El segundo son las que son modificadas por software. Las variables que serán modificadas en le ISR deben ser declaradas con el modificador volatile para indicar que pueden cambiar en cualquier momento. Hacer que los ISR sean lo más rápido posible. Los PICs de gama baja no admiten interrupciones, solo RESET. Se deshabilita la interrupción global cuando se detecta una.