INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRES TUXTLA

DIVISIÓN DE INGENIERÍA MECATRÓNICA

MATERIA:
Microcontroladores

PROFESOR:
Ing. Juan Merlín Chontal

GRUPO:
711-A (Séptimo semestre)

PERIODO ESCOLAR:
Agosto 2024 - diciembre 2024.

ACTIVIDAD:
Actividades unidad 3

ALUMNOS:
Jaden Casanova Gonzalez
Brando Coto Coto
Francisco Eduardo Azamar
Luis Javier Gómez Oliveros

San Andrés Tuxtla. Veracruz a 13 de noviembre del 2024

INVESTIGACIÓN.

Índice

3.1 Concepto de interrupción en un microcontrolador. 5

3.2 Manejo de interrupciones 6

3.2.1 Tipos de interrupciones. 7

3.2.2 Los vectores de interrupción. 7

3.2.3 Acciones del microcontrolador para el tratamiento de las interrupciones 9

3.2.4 Características de la rutina manejadora de interrupción. 11

3.3 Las interrupciones externas. 13

3.3.1 Características y configuración. 13

3.3.2 Programación y uso. 15

3.4 Fuentes internas de interrupción 17

3.4.1 De los Temporizadores y Contadores. 17

3.4.2 Del convertidor analógico digital. 17

3.4.3 De la comunicación serial (USART, SPI, TWI, etc.) 18

3.4.4 Del comparador analógico 18

Introducción

Las interrupciones representan una característica fundamental que permite para el ámbito de la electrónica una gestión eficiente de eventos y tareas en tiempo real, por su parte un microcontrolador, al ser un dispositivo importante e involucrado en innumerables dispositivos electrónicos, necesita reaccionar a diversos estímulos de manera rápida y precisa, las interrupciones nos permitirán que el microcontrolador interrumpa su flujo de ejecución normal para atender eventos críticos, optimizando así el rendimiento y la capacidad de respuesta del sistema.

En este caso exploraremos diversos aspectos del manejo de interrupciones en microcontroladores. definiendo el concepto de interrupción y su importancia en el contexto del control de procesos y la interacción con el entorno para poder detallar cómo se gestionan estas interrupciones, incluyendo los diferentes tipos que existen y los vectores de interrupción que se utilizan para direccionar el flujo del programa hacia las rutinas específicas de manejo de cada evento.

Con ello esperamos poder proporcionar una visión comprensiva sobre el funcionamiento y la importancia de las interrupciones en microcontroladores, resaltando su papel vital en el diseño y desarrollo de sistemas embebidos que responden a un entorno dinámico.

3.1 Concepto de interrupción en un microcontrolador.

Las interrupciones son eventos que hacen que el microcontrolador PIC deje de realizar la tarea actual y pase a efectuar otra actividad, y al finalizar la segunda actividad retorna a la primera y continúa a partir del punto donde se produjo la interrupción, esto permite que un solo microcontrolador ejecute varias tareas (no exactamente al mismo tiempo) dependiendo del evento que desencadene la interrupción, dichos dispositivos tienen desde 10 hasta 15 fuentes de interrupción dependiendo del tipo específico de PIC, el manejo de las interrupciones se programa por medio de registros especiales que controlan el comportamiento del microcontrolador bajo determinadas circunstancias.

El microcontrolador PIC16F88 tiene hasta 12 fuentes de interrupciones, el microcontrolador PIC16F628A tiene 10 y el microcontrolador PIC16F877A tiene 15.

El registro INTCON (Interrupt Control Register) es un registro fundamental en muchos microcontroladores PIC, utilizado para gestionar las interrupciones, este registro permite habilitar, deshabilitar y manejar el estado de las interrupciones, facilitando la programación de eventos asíncronos y el control del flujo de ejecución del programa. A continuación se describen las funciones y los bits principales que componen el registro INTCON en un microcontrolador típico, como el PIC16F84A.

REGISTER 2-3: INTCON: INTERRUPT CONTROL REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INT0IE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

3.2 Manejo de interrupciones

El manejo de interrupciones en un microcontrolador implica la capacidad de responder a eventos externos o internos de manera eficiente, interrumpiendo la ejecución normal del programa y permitiendo que el microcontrolador ejecute una rutina específica para atender la interrupción. A continuación, se abordan los subtemas relevantes para una comprensión profunda de este concepto

3.2.1 Tipos de interrupciones.

En los microcontroladores hay dos tipos de interrupciones, el primer tipo corresponde a eventos externos que generan un estado lógico (cambio de voltaje) y también por transición como un pulso electrónico de disparo (triggered), la transición se detecta por flanco de subida en la señal periódica (de bajo hacia alto, es decir: LOW → HIGH) o por flanco de caída (HIGH → LOW); esto habilita una bandera de interrupción (interrupt flag), dependiendo de la prioridad de interrupción, el contador del programa (program counter) toma la dirección de memoria de la tabla de vectores y salta a la localidad de memoria correspondiente donde se encuentra la rutina de la interrupción solicitada (interrupt handling routine). De manera autom´atica, por hardware se limpia la bandera de interrupción (interrupt flag); también se puede limpiar las banderas de interrupción por software, ya que tienen asociados sus respectivos bits de habilitación en el registro de estado (status register). Similarmente, si más solicitudes de interrupción ocurren mientras se encuentra en proceso alguna interrupción, permanecerán en espera por orden de prioridad. Cuando el contador de programa sale de una interrupción, retornará al programa principal y ejecutar a una o más instrucciones antes de atender alguna interrupción pendiente. El segundo tipo de interrupciones corresponden a las interrupciones que pueden ser cambiadas o reasignadas por software en los pins del microcontrolador, estas interrupciones no necesariamente tienen banderas de interrupción.

3.2.2 Los vectores de interrupción.

Un vector de interrupción es un array que contiene apuntadores a las localizaciones dentro de la RAM (memoria). La tabla de vectores de interrupción en él se indexa y acceden a través de su número de interrupción, una vez que se ha indicado el número de interrupción y en algunos casos realizado algunos cálculos para acceder a las posiciones de memoria donde se encuentra el vector involucrado a que se hace mención en la llamada a la interrupción. Una vez identificado la interrupción se procede a entregar el control del programa mediante un salto para la ejecución de las rutinas a la cual apunta el registro que contiene el dicho apuntador. Una vez que se ha terminado la ejecución del código al cual fue apuntado, el control es retornado al programa llamador (que ha generado la interrupción).

Los Microprocesadores en muchos casos incluyen instrucciones para gestionar las interrupciones internamente, es decir muchos de los valores son incluidos por el mismo sistema de hardware o software del sistema operativo, y mediante las llamadas a interrupciones se pueden ejecutar y tener acceso a las mismas.

Otra posibilidad constituye el hecho de que nosotros mismos podamos gestionar el acceso y ejecución de tales rutinas incluidas como apuntadores dentro del propio vector de interrupciones.

Si se llama incluye las llamadas a interrupciones, el mismo microprocesador gestiona las llamadas y retornos, incluyendo salvar y restaurar los registros internos del microprocesador que son necesarios para la llamada y retorno de las instrucciones que se van ejecutando.

Para el caso de que se necesite gestionar la tabla de interrupciones por cuenta propia se deben guardar y restaurar los registros (mediante la colocación en la pila los valores de registro y volver a restaurarlos quitando los valores anteriores para continuar con la ejecución normal del programa) antes y después de las llamadas provocadas por el salto de la ejecución de las interrupciones y su posterior retorno a la ejecución normal de las sentencias en el programa llamador.

Son por lo regular vectores de 32 bits, que están segmentadas en bloques de 16 bits cada una donde la parte más baja de la memoria se denomina Lo-Word y la parte alta del mismo se denomina Hi-word, (Word significa una palabra que es un bloque de 16 bits o 2 bytes, también se puede segmentar un Word en Byte-Lo y Byte-Hi respectivamente) pero no se los ve directamente como Word, sino que se los ve como apuntadores ya que cumplen esa función y específicamente FAR POINTERs, son apuntadores lejanos!!

3.2.3 Acciones del microcontrolador para el tratamiento de las interrupciones

El tratamiento de las interrupciones es un proceso fundamental en la operación de los microcontroladores, permitiendo una respuesta rápida a eventos críticos. Cuando ocurre una interrupción, el microcontrolador sigue un conjunto de acciones específicas para manejarla de manera eficiente. Estas acciones son esenciales para asegurar que el sistema funcione correctamente y que la ejecución del programa principal pueda reanudarse sin errores. A continuación, se describen las etapas del tratamiento de las interrupciones.

1. Detección de la Interrupción

El microcontrolador monitorea continuamente los registros de interrupción para detectar si alguna bandera de interrupción se ha activado. Esto implica verificar las condiciones definidas por las interrupciones externas e internas configuradas previamente. Cuando se cumple la condición para una interrupción, la bandera correspondiente se establece.

2. Interrupción del Flujo de Ejecución

Una vez que se detecta una interrupción, el microcontrolador interrumpe la ejecución normal del programa. Esta interrupción se produce de manera que el microcontrolador guarda el contexto actual, lo que incluye el contador de programa (PC) y otros registros necesarios para que el programa principal pueda reanudarse posteriormente.

3. Almacenamiento del Estado del Programa

Antes de saltar al vector de interrupción, el microcontrolador almacena el estado del programa actual en la pila (stack) o en un área de memoria designada. Este almacenamiento puede incluir:

- El valor del contador de programa (PC), que indica la instrucción que se estaba ejecutando.
- Los registros de trabajo y de estado que puedan ser necesarios para la reanudación posterior.

4. Salto al Vector de Interrupción

El microcontrolador utiliza el vector de interrupción correspondiente para determinar a qué dirección de memoria debe saltar. Cada tipo de interrupción tiene un vector específico que apunta a la rutina de servicio de interrupción (ISR) asociada. Esta rutina contiene el código que maneja el evento que provocó la interrupción.

5. Ejecución de la Rutina de Servicio de Interrupción (ISR)

La ISR es una función diseñada para manejar la interrupción. Durante su ejecución, la ISR puede realizar varias tareas, tales como:

- Procesar datos que llegaron a través de una entrada.
- Actualizar variables o estados del sistema.
- Reiniciar temporizadores.
- Enviar señales a dispositivos periféricos.

Es importante que la ISR sea breve y eficiente, ya que un tiempo prolongado en la ISR puede interferir con el manejo de otras interrupciones.

6. Restauración del Estado del Programa

Una vez que la ISR ha finalizado, el microcontrolador debe restaurar el estado original del programa. Esto implica recuperar el valor del contador de programa (PC) y otros registros que se guardaron antes de la interrupción. La restauración se realiza utilizando instrucciones específicas para volver a la ejecución normal del programa.

7. Reinicio de las Banderas de Interrupción

Después de ejecutar la ISR, es crucial que las banderas de interrupción asociadas se limpien (reset). Esto se hace para evitar que el microcontrolador entre nuevamente en la misma ISR de manera incontrolada. Si la bandera no se reinicia, la interrupción podría ser tratada de nuevo inmediatamente, lo que puede resultar en un bucle de interrupciones.

3.2.4 Características de la rutina manejadora de interrupción.

Las rutinas manejadoras de interrupción (ISR) son funciones críticas en la programación de microcontroladores, diseñadas para responder a eventos de interrupción de manera rápida y eficiente. Para garantizar que el sistema funcione correctamente y para maximizar la eficiencia del proceso, las ISR deben cumplir con ciertas características y buenas prácticas. A continuación se describen estas características:

1. Brevedad

Las ISR deben ser lo más cortas y concisas posible. Esto es crucial porque, durante la ejecución de una ISR, el microcontrolador no puede atender otras interrupciones. Una ISR prolongada puede causar un retraso en la respuesta a otras interrupciones y afectar el rendimiento general del sistema. Las rutinas deben realizar solo las tareas esenciales, como el procesamiento inmediato de datos o la actualización de estados.

2. No Uso de Funciones de Llamada

En general, las ISR no deben utilizar funciones de llamada (subrutinas) que requieran un contexto de pila adicional. Esto se debe a que la pila puede estar ocupada por el contexto del programa principal y la llamada a una función dentro de una ISR puede generar un desbordamiento de pila o comportamientos inesperados. En su lugar, se deben incluir solo instrucciones simples y directas.

3. Manejo de Banderas de Interrupción

Las ISR deben manejar adecuadamente las banderas de interrupción. Al inicio de la ISR, se debe verificar qué condición de interrupción ha causado el salto y, al final de la rutina, es esencial restablecer la bandera de interrupción correspondiente para evitar que la misma interrupción se procese nuevamente de manera incontrolada. Esto asegura que el sistema no quede atrapado en un bucle de interrupciones.

4. Acceso a Variables Globales

Las ISR pueden acceder y modificar variables globales. Sin embargo, es importante tener cuidado al hacerlo, ya que el acceso simultáneo desde el programa principal y la ISR puede provocar

condiciones de carrera. Para evitar problemas, se recomienda usar técnicas de exclusión mutua (mutex) o desactivar temporalmente las interrupciones al modificar variables críticas.

5. Uso de Registro de Estado

Algunas ISRs utilizan registros de estado para almacenar información sobre el estado actual del sistema. Esto puede ser útil para recuperar el contexto original al salir de la ISR y asegurar que la ejecución del programa principal continúe sin problemas.

6. Evitar Retardos o Esperas

Las ISR no deben incluir retardos o esperas (delays). Esto podría bloquear el microcontrolador y afectar su capacidad para manejar otras interrupciones. En lugar de eso, cualquier funcionalidad que requiera tiempo, como la espera de una señal o el procesamiento extenso de datos, debe ser realizada en el programa principal.

7. Compatibilidad con Otros Eventos

Las ISR deben ser diseñadas para ser compatibles con otros eventos de interrupción. Es posible que ocurran múltiples interrupciones en un corto período de tiempo, por lo que es importante que la ISR no interfiera con el manejo de otras interrupciones y que el sistema pueda gestionar múltiples eventos de manera eficiente.

3.3 Las interrupciones externas.

Las interrupciones externas sirven para detectar un estado lógico o un cambio de estado en alguna de las terminales de entrada de un microcontrolador, con su uso se evita un sondeo continuo en la terminal de interés. Son útiles para monitorear interruptores, botones o sensores con salida a relevador. En la tabla 4 se describen las interrupciones externas existentes en los AVR bajo estudio, en el ATMega8 se tienen 2 fuentes y en el ATMega16 son 3.

Tabla 4.1 Interrupciones externas y su ubicación en MCUs con encapsulado PDIP

ATMega8			ATMega16		
Fuente	Ubicación	Terminal	Fuente	Ubicación	Terminal
INT0	PD2	4	INT0	PD2	16
INT1	PD3	5	INT1	PD3	17
			INT2	PB2	3

3.3.1 Características y configuración.

Las interrupciones externas pueden configurarse para detectar un nivel bajo de voltaje o una transición, ya sea por un flanco de subida o de bajada. Con excepción de INT2, que sólo puede activarse por flancos. Las interrupciones pueden generarse aun cuando sus respectivas terminales sean configuradas como salidas. Las transiciones en INT0/INT1 requieren de la señal de reloj destinada a los módulos de los recursos (clkI/O, sección 2.8) para producir una interrupción, esta señal de reloj es anulada en la mayoría de los modos de bajo consumo (sección 2.9). Por el contrario, un nivel bajo en INT0/INT1 y las transiciones en INT2 no requieren de una señal de reloj para producir una interrupción, puede decirse que son eventos asíncronos, por lo que éstos son adecuados para despertar al microcontrolador, sin importar el modo de reposo.

Configuración.

La configuración de INT0 e INT1 se define en el registro MCUCR (MCU Control Register), los 4 bits más significativos de este registro están relacionados con los modos de bajo consumo de energía y fueron descritos en la sección 2.9, los 4 bits menos significativos son:

	7	6	5	4	3	2	1	0	
0x35	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	**MCUCR**

Bits 3 y 2 – ISC1[1:0]: Para configurar el sentido de INT1 (ISC, Interrupt Sense Control)

Definen el tipo de evento que genera la interrupción externa 1.

Bits 1 y 0 – ISC0[1:0]: Para configurar el sentido de INT0

Definen el tipo de evento que genera la interrupción externa 0.

En la tabla 4.2 se muestran los eventos que generan estas interrupciones, de acuerdo con el valor de los bits de configuración.

Tabla 4.2 Configuración del sentido de las interrupciones externas 0 y 1

ISCx1	ISCx0	Activación de la Interrupción
0	0	Por un nivel bajo de voltaje en INTx
0	1	Por cualquier cambio lógico en INTx
1	0	Por un flanco de bajada en INTx
1	1	Por un flanco de subida en INTx

x puede ser 0 ó 1

La configuración de INT2 se define con el bit ISC2 ubicado en la posición 6 del registro MCUCSR (MCU Control and Status Register).

	7	6	5	4	3	2	1	0	
0x34	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	**MCUCSR**

En la tabla 4.3 se muestran las transiciones que generan la interrupción externa 2, en función del bit ISC2.

Tabla 4.3 Configuración del sentido de la interrupción externa 2

ISC2	Activación de la Interrupción
0	Por un flanco de bajada en INT2
1	Por un flanco de subida en INT2

Dado que la interrupción 2 es asíncrona, se requiere que el pulso generador del evento tenga una duración mínima de 50 nS para que pueda ser detectado.

3.3.2 Programación y uso.

Cualquier interrupción va a producirse sólo si se activó al habilitador global de interrupciones y al habilitador individual de la interrupción de interés. El habilitador global es el bit I, ubicado en la posición 7 del registro de Estado (SREG, sección 2.4.1.1).

Los habilitadores individuales de las interrupciones externas se encuentran en el registro general para el control de interrupciones (GICR, General Interrupt Control Register), correspondiendo con los 3 bits más significativos de GICR:

	7	6	5	4	3	2	1	0	
0x3B	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	**GICR**

- Bit 7 – INT1: Habilitador individual de la interrupción externa 1
- Bit 6 – INT0: Habilitador individual de la interrupción externa 0
- Bit 5 – INT2: Habilitador individual de la interrupción externa 2
  No está disponible en un ATMega8.
- Bits 4 al 2 – No están implementados
- Bits 1 y 0 – No están relacionados con las interrupciones externas

El estado de las interrupciones externas se refleja en el registro general de banderas de interrupción (GIFR, General Interrupt Flag Register), el cual incluye una bandera por interrupción, estas banderas corresponden con los 3 bits más significativos de GIFR:

- Bit 7 – INTF1: Bandera de la interrupción externa 1
- Bit 6 – INTF0: Bandera de la interrupción externa 0
- Bit 5 – INTF2: Bandera de la interrupción externa 2
  No está disponible en un ATMega8.
- Bits 4 al 0 – No están implementados

Las banderas se ponen en alto si el habilitador global y los habilitadores individuales están activados y ocurre el evento definido por los bits de configuración. La puesta en alto de una de estas banderas es lo que produce la interrupción, dando lugar a los 122 procedimientos descritos en la sección 2.6.1, la bandera se limpia automáticamente por hardware cuando se concluye con la atención a la interrupción. No es necesario evaluar las banderas por software, puesto que se tiene un vector diferente para cada evento.

3.4 Fuentes internas de interrupción

Las fuentes internas de interrupción son señales o eventos que se generan dentro del propio microcontrolador y que pueden activar una interrupción, estas se encuentran relacionadas con los diferentes módulos y periféricos integrados en el microcontrolador y permiten que el sistema responda automáticamente a ciertos eventos internos, sin necesidad de intervención externa, resultan fundamentales para el funcionamiento eficiente de sistemas embebidos, ya que permiten que el microcontrolador gestione de manera automática tareas críticas o eventos importantes.

3.4.1 De los Temporizadores y Contadores.

Los temporizadores y contadores en un microcontrolador son periféricos que permiten medir tiempos precisos o contar eventos de manera automática, estos dispositivos funcionan incrementando un registro con cada ciclo de reloj o cada evento de entrada, al momento en que alcanzan un valor específico o se desbordan (es decir, vuelven a cero después de haber alcanzado su valor máximo), se puede activar una interrupción que avise al microcontrolador de que un intervalo de tiempo ha transcurrido o que se ha contado un número determinado de eventos.

Esto es útil en aplicaciones donde se requiere una sincronización precisa, como en la generación de señales de reloj, el control de ciclos PWM (modulación por ancho de pulso), y la ejecución de tareas periódicas, como la adquisición de datos en intervalos de tiempo específicos, dentro de este contexto, los temporizadores y contadores funcionan en segundo plano y sólo requieren la atención del microcontrolador cuando una tarea precisa ejecutarse, optimizando así los recursos del sistema.

3.4.2 Del convertidor analógico digital.

Los convertidores analógico-digital (ADC) son módulos internos de los microcontroladores que permiten convertir señales analógicas (continuas) en valores digitales discretos, que pueden ser procesados por el microcontrolador, para sistemas embebidos, los ADC son esenciales para procesar datos provenientes de sensores analógicos, como sensores de temperatura, luz, o presión.

Una vez que se inicia la conversión en un ADC, el microcontrolador puede seguir ejecutando otras tareas, cuando finaliza la conversión, el módulo ADC puede generar una interrupción que le avisa al microcontrolador que el dato digital ya está listo, esto permitirá al sistema reaccionar solo cuando la conversión ha terminado, evitando la necesidad de monitorear continuamente el proceso y mejorando el rendimiento, para sistemas de adquisición de datos, esta capacidad de interrupción asegura una respuesta rápida y eficiente a las lecturas analógicas sin ralentizar el programa principal.

3.4.3 De la comunicación serial (USART, SPI, TWI, etc.)

Los microcontroladores a menudo deben comunicarse con otros dispositivos, como sensores, controladores, o dispositivos de almacenamiento, mediante protocolos de comunicación serial, los módulos de comunicación serial como USART (Transmisión y Recepción Asíncrona Universal), SPI (Interfaz Periférica Serial) e I2C (Inter-Integrated Circuit) facilitan esta interacción, cada protocolo tiene sus propias ventajas y aplicaciones específicas, desde la transmisión de datos a alta velocidad (SPI) hasta la transmisión de datos en múltiples dispositivos en un solo bus (I2C).

Cuando se transmite o recibe un byte de datos a través de estos módulos, el microcontrolador no necesita monitorear constantemente el proceso, en su lugar el módulo genera una interrupción cada vez que se completa la transferencia, permitiendo que el microcontrolador procese los datos en cuanto están disponibles, lo que resulta esencial para sistemas que requieren un intercambio de datos continuo, liberando al microcontrolador para realizar otras tareas mientras espera por los datos entrantes o para enviar otros.

3.4.4 Del comparador analógico

El comparador analógico es un módulo que permite comparar dos voltajes de entrada y determinar cuál es mayor o si uno de ellos ha superado un umbral predeterminado, al momento en que este

cambio ocurre el comparador puede generar una interrupción que avise al microcontrolador sobre la condición, los comparadores analógicos son útiles en aplicaciones que requieren monitoreo en tiempo real, como sistemas de protección de voltaje o en controladores de potencia.

Por ejemplo, en un sistema de monitoreo de batería, el comparador analógico puede generar una interrupción cuando el voltaje de la batería cae por debajo de un nivel crítico, activando un proceso de emergencia o apagando el sistema para evitar daños. dicha función permitirá al sistema responder de inmediato sin requerir que el microcontrolador verifique constantemente el voltaje, optimizando la respuesta a eventos críticos.

Conclusión.

Las interrupciones son un pilar clave en diversos sistemas, aportando una capacidad de respuesta rápida y efectiva que permite al microcontrolador interactuar eficazmente con su entorno y procesar eventos en tiempo real, al analizar sus distintos tipos y la manera en que son gestionadas, se demuestra que las interrupciones optimizan el rendimiento general del sistema, permitiendo que este atienda solo aquellos eventos prioritarios y libere recursos para otras tareas secundarias cuando sea posible, por otra parte los vectores de interrupción, que dirigen el flujo de ejecución hacia las rutinas específicas para cada evento, ilustran cómo el microcontrolador maneja la multitarea de manera estructurada y precisa, permitiendo un control preciso sobre los procesos que requieren sincronización o monitoreo continuo.

En definitiva, comprender y manejar adecuadamente las interrupciones permite maximizar la eficiencia y confiabilidad del microcontrolador, haciendo que los sistemas embebidos no solo operen con mayor rapidez y menor consumo de recursos, sino que también sean capaces de adaptarse dinámicamente a diferentes condiciones. Esto convierte a las interrupciones en una herramienta indispensable para el desarrollo de dispositivos electrónicos modernos, eficientes y adaptables que operan en un entorno de cambio constante y que requieren una respuesta inmediata y confiable.

Referencias

1. Microchip Technology Inc. (2023). PIC® Microcontroller Interrupts. Microchip Developer Help.
2. Morris, J. (2014). Microcontroller Theory and Applications with the PIC18F. Pearson.
3. Velasco, N. (2016). Microcontroladores. línea]. Disponible: http://www. controlesdigitales. com/Libro_Felipe_Santiago/03_Cap_1_2_3.pdf [Último acceso: 03 11 2021].
4. de ATMEL, M. A. (2013). Tarjeta de Adquisición de Datos con interfaz USB, empleando al (Doctoral dissertation, UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA).
5. ¿Cuáles son algunos ejemplos de interrupciones y temporizadores en un microcontrolador que ha utilizado o de los que ha aprendido? (2023, 15 marzo). www.linkedin.com. https://es.linkedin.com/advice/1/what-some-examples-interrupts-timers-microcontroller?lang=es
6. ENGINEERING THE WORLD FROM PARAGUAY. (2011, 31 marzo). Sobrevolando los vectores de interrupciones. Rama Estudiantil del IEEE de la UCSA. https://ramaucsa.wordpress.com/2011/03/31/2304/
7. Peatman, J. B. (1998). Design with PIC microcontrollers. Pearson Education India.
8. Lucio, D. (2010). Microcontroller Programming and Interfacing with Texas Instruments MSP430FR2433 and MSP430FR5994 (3rd ed.). Elsevier
9. Huang, H. (2005). The HCS12 / 9S12: An Introduction to Software and Hardware Interfacing. Thomson Delmar Learning.
10. Reyes, F., & Cid, J. (2015). Arduino: aplicaciones en robótica, mecatrónica e ingenierías. *España: Marcombo*.
11. Interrupciones - Wikitronica. (s. f.). http://wikitronica.labc.usb.ve/index.php/Interrupciones
12. Rangel, P. (s. f.). MANEJO DE INTERRUPCIONES EN C. https://profesor-rangel.blogspot.com/p/manejo-de-interrupciones.html
13. TECmikro Ecuador. (s. f.). Interrupciones de los microcontroladores PIC. https://tecmikro.com/content/68-interrupciones-microcontroladores-pic#:~:text=Las%20interrupciones%20permiten%20que%20un,del%20tipo%20espec%C3%ADfico%20de%20PIC.

EXPOSICIÓN.

Foto de la exposición

SIMULACIÓN DE CIRCUITO EN PROTEUS.

Introducción.

En la actualidad ya se utiliza mas la informática, para el desarrollo de ciertos sistemas a utilizar, porque con el avance de la tecnología las grandes empresas necesitan actualizarse para estar en desarrollo con su competencia para lo cual se utilizan ciertos softwares para brindar un mejor funcionamiento en el proceso de fabricación, además de utilizar interrupciones de microcontroladores que lo ayudaran a realizar ciertas tareas.

Además, para detectar un cambio de estado en un pin de entrada, se emplean instrucciones para consultar de manera repetida el estado de la entrada.

Lo cual se realiza continuamente con tiempo de espera entre consultas (utilizando el comando delay).

El procesador consume energía adicional debido a la pregunta que realiza de manera constante. Si se requiere de atención inmediata al pedido, no se podrá atender, ya que se debe esperar que una instrucción consulte si se realiza el pedido. Si el pulso es de corta duración, o si el procesador está ocupado en otra tarea, es posible que no se detecte el cambio.

A continuación, se presenta el código que nos ayudara a simular el circuito que se presentara más delante. Esto con la finalidad de conocer cómo se desarrollan los tiempos de temporización mediante la interrupción por desbordamiento.

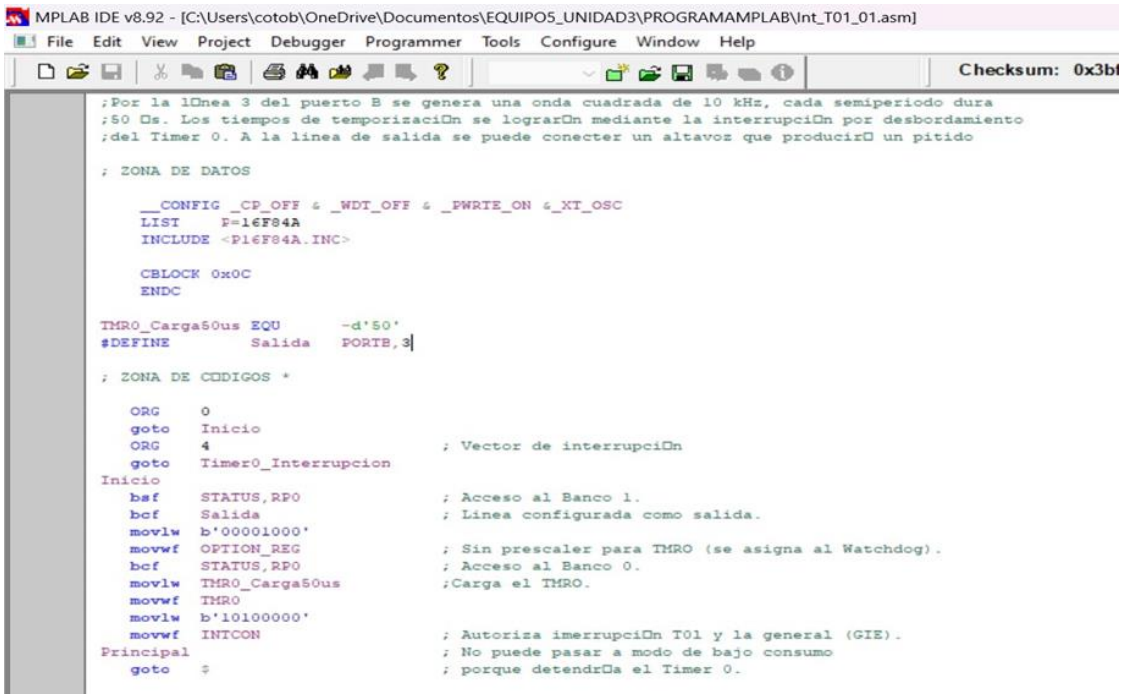
Código.

```
;Por la línea 3 del puerto B se genera una onda cuadrada de 10 kHz, cada semiperiodo dura
;50 µs. Los tiempos de temporización se lograrán mediante la interrupción por desbordamiento
;del Timer 0. A la linea de salida se puede conecter un altavoz que producirá un pitido

; ZONA DE DATOS

    __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON &_XT_OSC
    LIST    P=16F84A
    INCLUDE <P16F84A.INC>

    CBLOCK 0x0C
    ENDC
TMR0_Carga50us EQU      -d'50'
#DEFINE        Salida   PORTB,3

; ZONA DE CÓDIGOS *
   ORG    0
   goto   Inicio
   ORG    4                      ; Vector de interrupción
   goto   Timer0_Interrupcion
Inicio
```

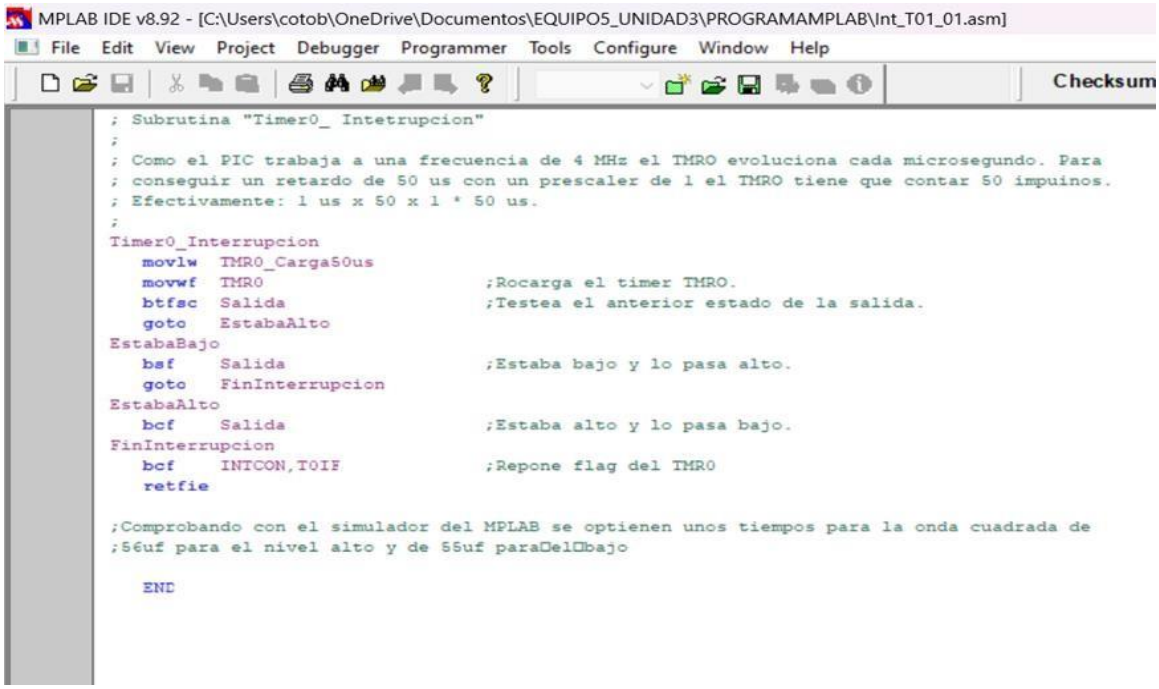
```
bsf    STATUS,RP0              ; Acceso al Banco 1.
   bcf    Salida                  ; Linea configurada como salida.
   movlw  b'00001000'
   movwf  OPTION_REG               ; Sin prescaler para TMRO (se asigna
al Watchdog).
   bcf    STATUS,RP0              ; Acceso al Banco 0.
   movlw  TMR0_Carga50us          ;Carga el TMRO.
   movwf  TMR0
   movlw  b'10100000'
   movwf  INTCON                  ; Autoriza imerrupción T0l y la general
(GIE).
Principal                         ; No puede pasar a modo de bajo consumo
   goto   $                        ; porque detendría el Timer 0.



; Subrutina "Timer0_ Intetrupcion"
;
; Como el PIC trabaja a una frecuencia de 4 MHz el TMRO evoluciona cada
microsegundo. Para
; conseguir un retardo de 50 us con un prescaler de 1 el TMRO tiene que
contar 50 impuinos.
; Efectivamente: 1 us x 50 x 1 * 50 us.
;
Timer0_Interrupcion
   movlw  TMR0_Carga50us
   movwf  TMR0                     ;Rocarga el timer TMRO.
   btfsc  Salida                  ;Testea el anterior estado de la salida.
```

```
   goto   EstabaAlto
EstabaBajo
   bsf    Salida                  ;Estaba bajo y lo pasa alto.
   goto   FinInterrupcion
EstabaAlto
   bcf    Salida                  ;Estaba alto y lo pasa bajo.
FinInterrupcion
   bcf    INTCON,T0IF             ;Repone flag del TMR0
   retfie

;Comprobando con el simulador del MPLAB se optienen unos tiempos para la
onda cuadrada de
;56uf para el nivel alto y de 55uf para el bajo

   END
```

Programa en lenguaje ensamblador.

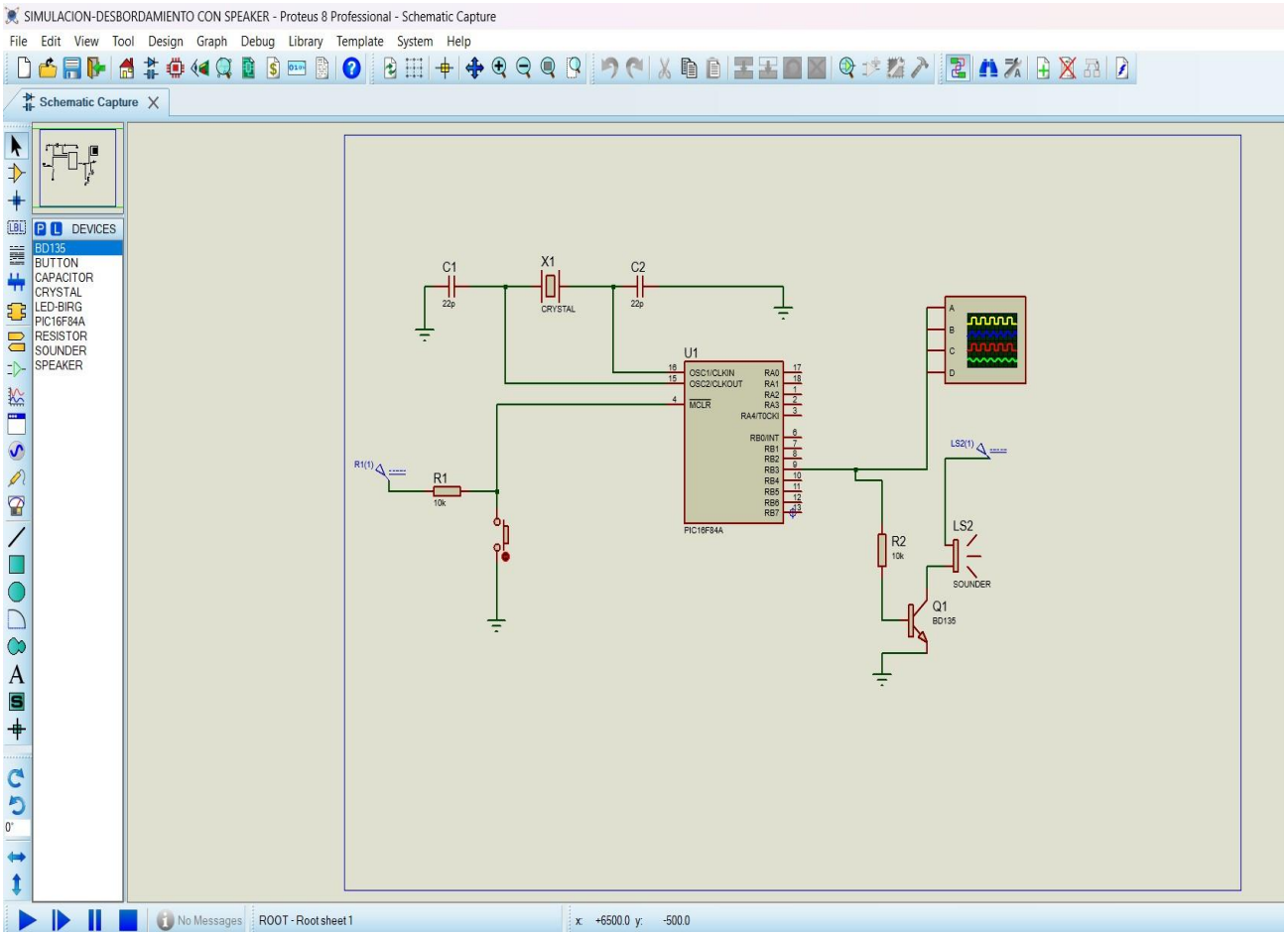


Se desarrolló el siguiente código con el PIC 16F84A con sus respectivos datos donde nos ayudara analizar el cambio que se hace en la interrupción por desbordamiento en cuanto el tiempo de temporización y cuando se le agrega un altavoz.

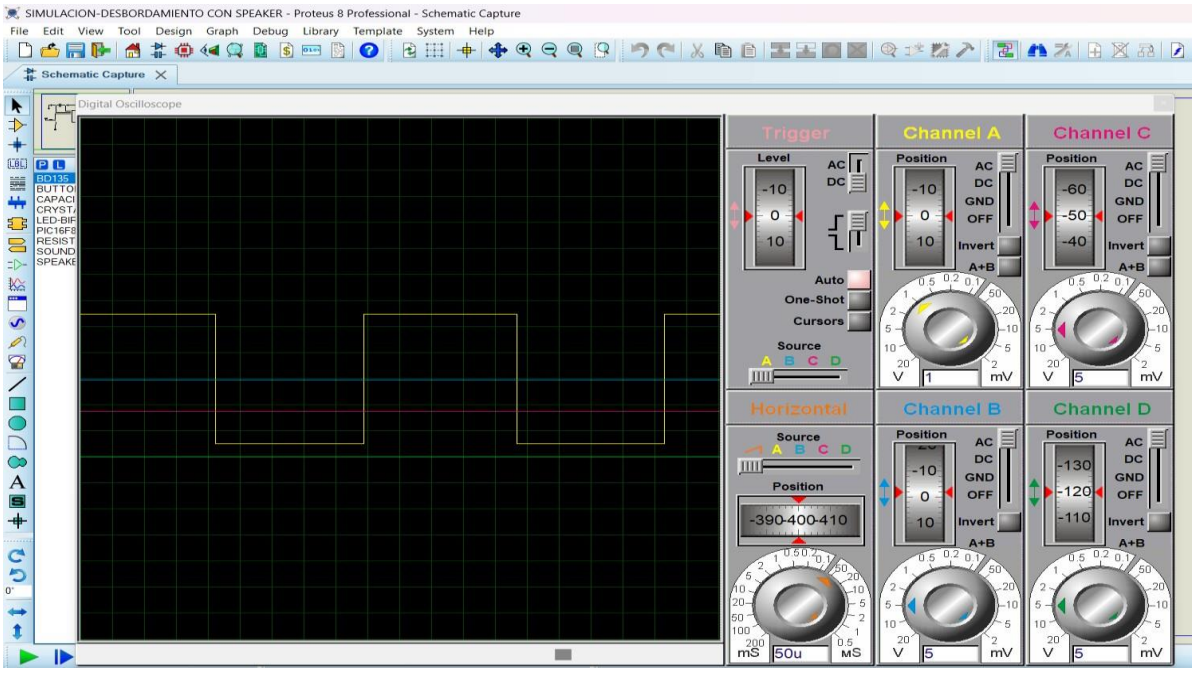


Con la simulación en MPLAB se obtienen unos tiempos para la onda cuadrada de 56 uf para el nivel alto y 55 uf para el nivel bajo.

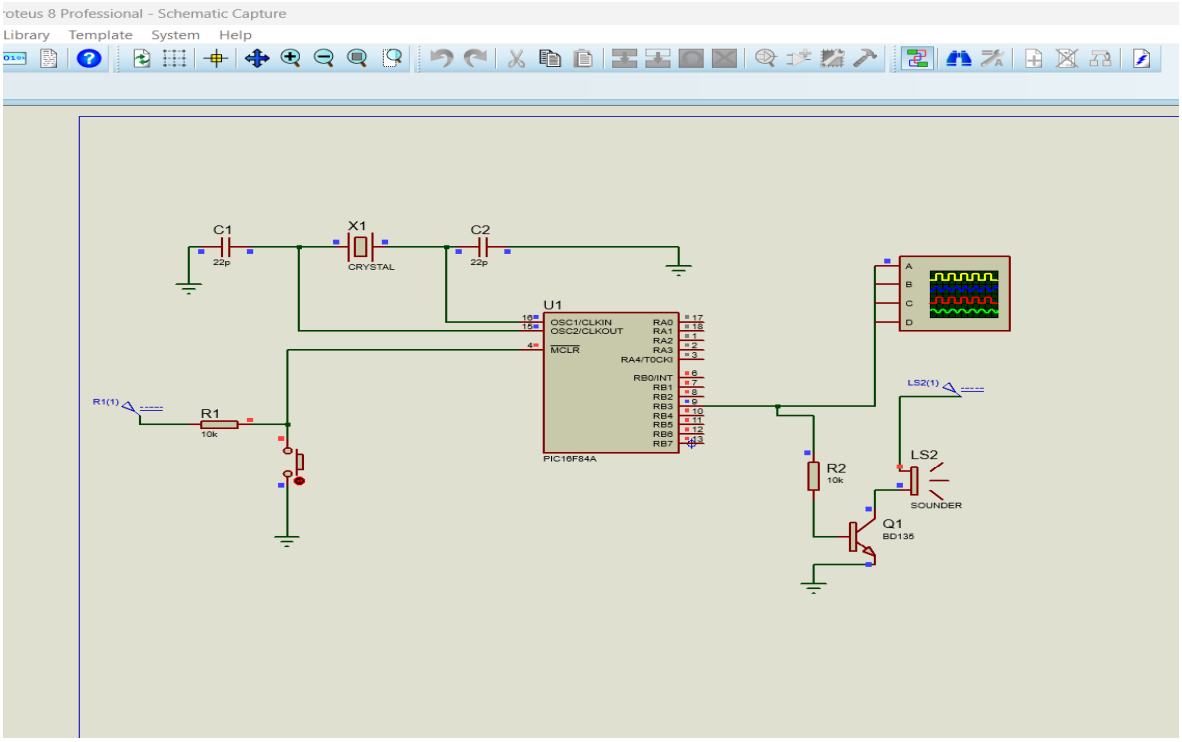
Circuito en proteus.



A continuación, se presenta la simulación del circuito del código que se presento con anterioridad. Esto en representación del desbordamiento con speaker. Nos muestra el comportamiento que ocurre cuando hay una interrupción de desbordamiento



Se muestra el digital Oscilloscope de la simulación del desbordamiento con speaker.



Se presenta el funcionamiento de la simulación cuando se cargó el código de interrupción por desbordamiento.

1. Cálculo de la frecuencia

- El microcontrolador PIC16F84A se ejecuta a una frecuencia de reloj de 4 MHz. Esto significa que cada ciclo de instrucción tarda 1 µs (ya que la frecuencia de instrucción es el cuarto de la frecuencia del oscilador principal).
- El temporizador TMR0 se configura para desbordar cada 50 µs. En cada desbordamiento del temporizador, el código de la rutina de interrupción cambia el estado de la línea de salida (de alto a bajo o viceversa), lo que forma un semiperiodo de la onda cuadrada.
- Como un ciclo completo de la onda cuadrada se compone de dos semiperíodos (uno alto y uno bajo), el período total es de 100 µs (50 µs + 50 µs).

2. Frecuencia de la Onda Cuadrada

La frecuencia de la onda cuadrada se calcula como el inverso del período:

$$F = \frac{1}{P} = \frac{1}{100\mu s} = 10,000Hz$$

3. Simulación y Verificación

Para confirmar que la onda es de 10 kHz:

- Realizamos una simulación de microcontroladores, como **MPLAB SIM** (de Microchip) o herramientas como **Proteus**, para simular el código y verificar visualmente el comportamiento de la señal.
- Un osciloscopio real podría medir la salida en RB3 si se implementa de manera eficiente, mostrando la onda cuadrada con la frecuencia esperada.

Para conocer la forma en que se desenvuelve el circuito es necesario que conozcamos dos definiciones muy importantes, el registro option e intcom.

4. Como variar la frecuencia de la onda cuadrada para que sea diferente de 10KHz

Para confirmar que la onda cuadrada generada es de 10 kHz, podemos realizar las siguientes comprobaciones:

Para variar la frecuencia de la onda cuadrada generada (por ejemplo, para obtener una frecuencia diferente de 10 kHz), puedes cambiar el tiempo de retardo configurado en el temporizador TMR0.

La frecuencia de la onda cuadrada está directamente relacionada con el tiempo que tarda el temporizador en provocar una interrupción. Actualmente, el temporizador se recarga cada 50 µs, lo que da una frecuencia de 10 kHz. Si cambiamos este valor, la frecuencia cambiará.

Registro OPTION.

El registro OPTION, o OPTION_REG en algunos microcontroladores como los de la familia PIC, es un registro especial de control que permite configurar ciertas opciones y características de hardware del microcontrolador. Aquí te explico en detalle cada una de sus funciones más comunes:

1. **Prescalers (Divisores de Frecuencia)**:
    - El registro OPTION se usa comúnmente para configurar los prescalers, que son divisores de frecuencia. Estos dividen la frecuencia de una señal de reloj para reducirla a un valor más manejable.
    - Los prescalers se usan en temporizadores o en el *Watchdog Timer (WDT)*, un temporizador que supervisa el sistema para reiniciarlo si detecta problemas.
    - Cambiar los valores del prescaler afecta la frecuencia de los temporizadores, y con esto puedes controlar la frecuencia de eventos de interrupción temporizados, como el encendido de LEDs, la activación de motores, o la captura de datos a intervalos precisos.
    - En microcontroladores, el registro OPTION puede configurar temporizadores, prescalers y fuentes de reloj, los cuales determinan la frecuencia de algunas señales internas o externas del microcontrolador.
    - Por ejemplo, al configurar un temporizador en un microcontrolador, podrías usar el registro OPTION para ajustar la frecuencia a la que se activa la señal de salida.

Luego, si esta señal se envía a un pin de salida, puedes usar un osciloscopio para observar y medir su frecuencia.

- Así, el OPTION influye en la frecuencia de señales generadas por el microcontrolador, pero solo indirectamente con respecto al osciloscopio.

2. **Selección del Flanco de Interrupción Externa:**

- El registro OPTION también puede definir el flanco del pulso (ascendente o descendente) que activará una interrupción externa. Esto es útil si deseas detectar eventos específicos en una señal, como el inicio o fin de un pulso.
- Esta configuración es importante cuando trabajas con sensores o dispositivos que emiten señales de pulso, ya que determina en qué momento exacto se genera la interrupción en función de la señal de entrada.

3. **Activación de Pull-ups en Entradas:**

- Algunos microcontroladores permiten activar o desactivar resistencias pull-up internas en los pines de entrada mediante el registro OPTION. Las resistencias pull-up son útiles cuando deseas asegurar que los pines de entrada tengan un valor conocido (alto o bajo) cuando no están conectados a ningún otro componente.
- Esto es común en dispositivos como botones, donde quieres evitar que un pin flote (es decir, que tenga valores aleatorios).

INTCOM (Interrupción de comunicación)

INTCOM, en el contexto de microcontroladores, hace referencia a las interrupciones de comunicación que se utilizan para manejar la transmisión y recepción de datos en protocolos serie, como UART, SPI e I2C. Cada uno de estos protocolos cuenta con sus propios tipos de interrupciones, permitiendo que el microcontrolador reaccione automáticamente cuando ocurre un evento de comunicación.

Principales Usos de intcom en Comunicación Serie

1. **Interrupción UART:**

- Transmisión y Recepción Asíncrona: En UART (Universal Asynchronous Receiver/Transmitter), el microcontrolador puede generar una interrupción cuando se recibe un dato (interrupt de recepción) o cuando el buffer de transmisión está listo para enviar otro dato (interrupt de transmisión).
- Aplicaciones: Es útil en comunicación serie para recibir datos de sensores o enviar información a una pantalla sin detener el flujo del programa principal.

2. **Interrupción SPI:**

- Transferencia de Datos en Full-Duplex: En el protocolo SPI (Serial Peripheral Interface), las interrupciones permiten gestionar el envío y la recepción simultánea de datos en comunicación full-duplex.
- Aplicaciones: SPI es muy común en la comunicación con periféricos de alta velocidad, como pantallas, memorias o sensores. Con una interrupción SPI, el microcontrolador puede manejar datos rápidamente sin interrumpir otras tareas.

3. **Interrupción I2C:**

- Sincronización Maestra-Esclava: En I2C, el intcom permite al microcontrolador responder a solicitudes de lectura o escritura de un dispositivo maestro o esclavo. Las interrupciones I2C pueden activarse en eventos específicos como el inicio o fin de transmisión.

- Aplicaciones: Muy usado en sistemas donde varios dispositivos (como sensores o expansores de pines) se comunican en un bus compartido. La interrupción asegura que el microcontrolador responda rápidamente sin afectar el código principal.

- OPTION: Configura aspectos de hardware relacionados con temporizadores y entradas. Es útil para controlar la frecuencia de temporizadores (prescalers), activar interrupciones en un flanco específico, o habilitar pull-ups en pines de entrada.
- intcom: Enfocado en la gestión de interrupciones de comunicación, facilita la transmisión y recepción de datos en UART, SPI, e I2C, sin interferir con el flujo del programa principal. Las interrupciones de comunicación permiten la gestión de datos de manera asíncrona y eficiente.

Conclusión.

Proteus es una aplicación que nos ayuda a desarrollar la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas, diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción.

En esta práctica nos ayudó a conocer la simulación de un circuito con respecto a el tema de la interrupción de desbordamiento. Por lo tanto, se presenta el funcionamiento de la simulación cuando se cargó el código de interrupción por desbordamiento.

Con la finalidad de comprender el comportamiento que desarrolla al tener una interrupción cuando esta en una tarea.

Como se sabe al realizar esta práctica y plasmarlo en el siguiente reporte, nos ayuda a comprender que el alumno obtuvo los conocimientos básicos de la unidad que se estudió para después ponerlo en práctica de acuerdo a las actividades proporcionadas.

CIRCUITO RELEVADORES CON LEDS

Fotos (evidencia)