Notas Técnicas de Uso y Aplicación

NT0011

1.11 ADC – CONVERSIÓN CONTINUA E INTERRUPCIONES
LECTURA DE UNA SEÑAL ANALÓGICA PROVENIENTE DE UN POTENCIÓMETRO

Preparado por: Rangel Alvarado
Estudiante Graduando de Lic. en Ing. Electromecánica
Universidad Tecnológica de Panamá
Panamá, Panamá
"e-mail": issaiass@cwpanama.net
"web site": http://www.geocities.com/issaiass/

ÍNDICE

1.11.1 Introducción *152*
1.11.2 ¿Qué es una Interrupción? *153*
1.11.3 Conversión Continua con Interrupción *155*
1.11.4 Diagrama de Flujo *156*
1.11.5 Código *158*
1.11.6 Simulador *164*
1.11.7 Conclusión *165*
1.11.8 Referencias *165*
1.11.9 Problemas Propuestos *165*

1.11.1 Introducción

La conversión analógica a digital es un gran recurso para captar señales del mundo real, pero en ciertas ocasiones se requiere una conversión continua, que asegure que el sensor o elemento a sensar es leído en momentos correctos. Para esto, se utilizan interrupciones, que son peticiones del sistema a atender situaciones externas al programa que se está ejecutando.

Esta nota abarca los aspectos como:

- Interrupciones: Las interrupciones, por lo general son procedimientos que se ejecutan por acciones del "hardware" externo y atender un evento inusual. Una programación "inteligente" incluye el uso de interrupciones que da mayor flexibilidad a programar.

- Conversión Continua: La conversión continua se utiliza para generar lecturas analógicas a digitales repetidamente así sin preocuparse de perdidas de tiempo para leer el convertidor.

- Programa Ejemplo: El programa ejemplo trabaja la interrupción del ADC y el potenciómetro para variar la rata de parpadeo del LED.

- Simulación: Se utilizan los puntos de ruptura ("breakpoints") para corroborar el funcionamiento del programa.

1.11.2 ¿Qué es una Interrupción?

En esta sección refiérase a la figura 100 y la tabla 46.

Una interrupción es una petición física de "hardware" para llamar la atención al CPU y atender un evento inusual. En otras palabras, un programa que está en ejecución pasa a otra sección, a ejecutar otra parte del programa más importante en ese momento y luego retorna al mismo punto donde fue interrumpido. Una interrupción no es lo mismo que una subrutina; la interrupción ocurre asincrónicamente (en cualquier momento), por otro lado, la subrutina tiene que esperar a ser ejecutada por el programa al pasar por ese segmento de código.

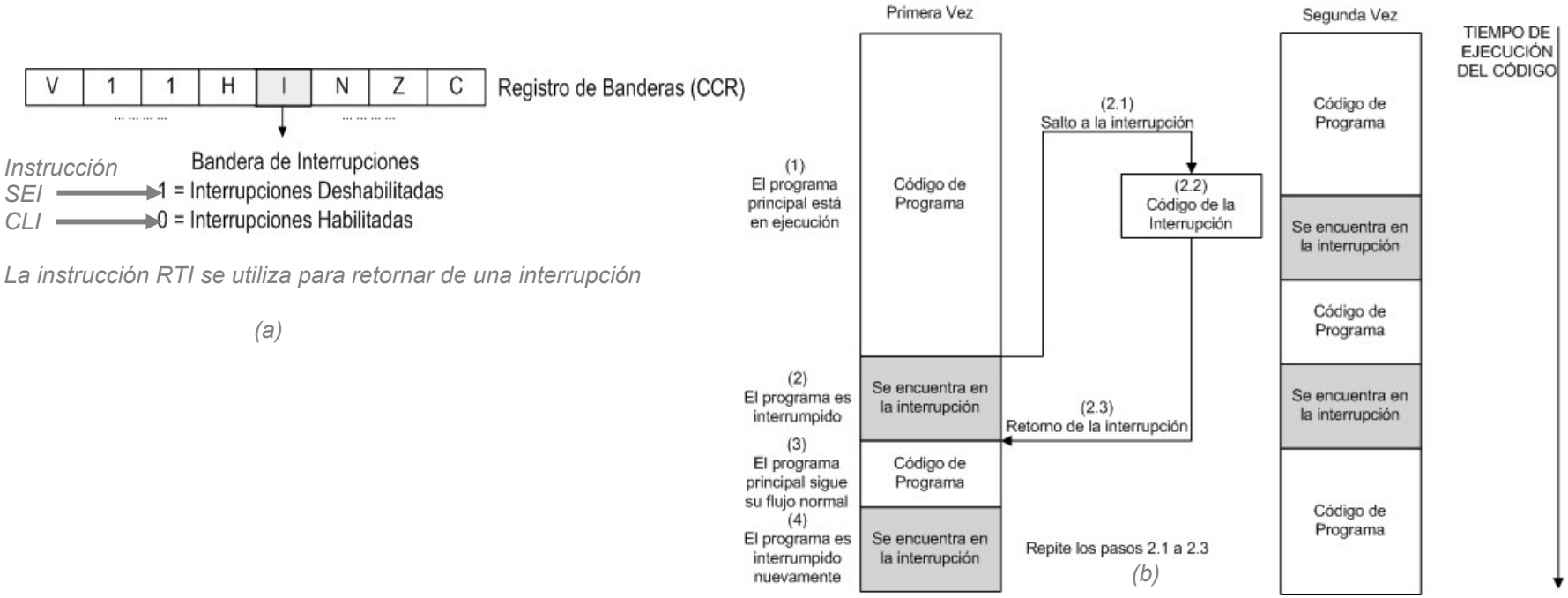


Figura 100. Manejo de una Interrupción. (a) Registro de Banderas. El bit "I", en el registro de banderas decide si las interrupciones serán o no procesadas. La instrucción SEI impone un uno (1) al bit "I" deshabilitando la interrupción (enmascarándola), mientras que la instrucción CLI impone un cero (0) al bit "I", habilitando las interrupciones.(b) Acto de una interrupción en el código de un programa. La interrupción, a diferencia de la subrutina, ocurre en cualquier momento para atender aquel código que es más prioritario en ese momento.

Existen dos tipos de interrupciones:

- Interrupciones de “hardware”: son interrupciones generadas por señales externas (pin IRQ) y módulos internos (temporizadores, ADC, LVI).
  Se pueden enmascarar, es decir, puede ocultarse el salto a la rutina de interrupción.

- Interrupciones de “software”: es una instrucción ejecutable a pesar del estado del bit *"I"* en el registro de banderas (CCR).
  No son enmascarables, es decir, no se puede ocultar su ejecución a la rutina de interrupción.

Tabla 46. Interrupciones y Máscaras del Microcontrolador JK1/JK3/JL3

Prioridad	*Vector*	*Bandera*	*Máscara*	*Dirección*	*Vector (Dirección)*
La más baja	IF15	COCO	AIEN	FFDE	Conversión completa ADC (Alta)
				FFDF	Conversión completa ADC (Baja)
	IF14	KEYF	IMASKK	FFF0	Teclado (Alta)
				FFF1	Teclado (Baja)
	IF13 a IF6	-----	-----	-----	No usado
	IF5	TOF	TOIE	FFF2	Sobreflujo del TIM (Alto)
				FFF3	Sobreflujo del TIM (Bajo)
	IF4	CH1F	CH1IE	FFF4	Canal 1 del TIM (Alto)
				FFF5	Canal 1 del TIM (Bajo)
	IF3	CH0F	CH0IE	FFF6	Canal 0 del TIM (Alto)
				FFF7	Canal 0 del TIM (Bajo)
	IF2			-----	No usado
	IF1	IRQF1	IMASK1	FFFA	Vector $\overline{\text{IRQ}}$ (Alto)
				FFFB	Vector $\overline{\text{IRQ}}$ (Bajo)
	-----	-----	-----	FFFC	Vector SWI (Alto)
				FFFD	Vector SWI (Bajo)
	-----	-----	-----	FFFE	Vector de Reinicio (Alto)
La más alta				FFFF	Vector de Reinicio (Bajo)

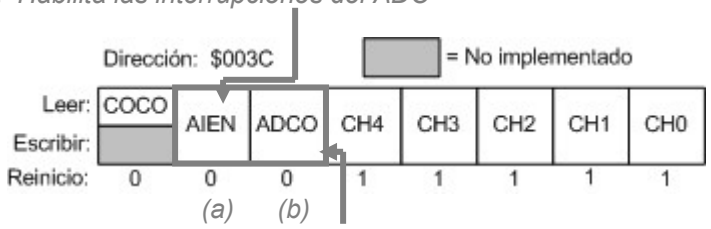
Nota: TIM = Módulo de Interfase por Temporizador
$\overline{\text{IRQ}}$ = Módulo de Interrupción Externa
SWI = Interrupción por “Software”

1.11.3 Conversión Continua con Interrupción

Para el programa de la sección 1.11.4, se utilizará una conversión continua con interrupciones. Para más información de los registros del ADC revistar la NT0012.

1.11.3.1 Registro de Control y Estado

AIEN = 1. Habilita las interrupciones del ADC



ADCO = 1. Habilita la conversión continua
Nota: Si AIEN = 1, el bit COCO siempre será cero (0) al ser leído

Figura 101. Bits de Interrupción y Conversión Continua del ADC. (a) AIEN. Habilita o deshabilita las interrupciones del convertidor analógico digital. (b) ADCO. Convierte continuamente y almacena en el registro ADR sin importar si se ha leído el mismo.

1.11.3.2 Declaración de una Interrupción y Petición de Interrupción

Para mayor información de la declaración de interrupciones, ver el fin del listado 10

La petición de interrupción es escrita de la forma:

```
<etiqueta de la interrupción>                    ; ADCINTL
        ---              ; Código de la interrupción      ---
        rti              ; retorno de la interrupción     rti
                                  (a)
```

Una interrupción se declara de la siguiente forma:

```
<etiqueta del vector interrupción parte alta>
; ADCINTH     equ $FFDE          ; Define el vector interrupción parte alta
        org <Dirección de Interrupción>          org ADCINTH
        dw <etiqueta de la interrupción>         dw ADCINTL
                                  (b)
```

Listado 8. Declaración y Petición de una Interrupción. (a) Petición de Interrupción. Debe tener una etiqueta vinculada a la interrupción deseada y un retorno a interrupción. (b) Declaración del vector de interrupción. Individualmente se declaran por el pseudo-operando "org", seguido del vector alto; dada la interrupción, la dirección del pseudo-operando "dw o fdb" es buscada automáticamente por el CPU del microcontrolador.

1.11.4 Diagrama de Flujo

El siguiente programa utiliza el potenciómetro para variar la rata de parpadeo de un LED conectado al puerto PTD7. El programa utiliza interrupciones del ADC y convierte continuamente el valor analógico del potenciómetro conectado a ADC7 (PTB7).

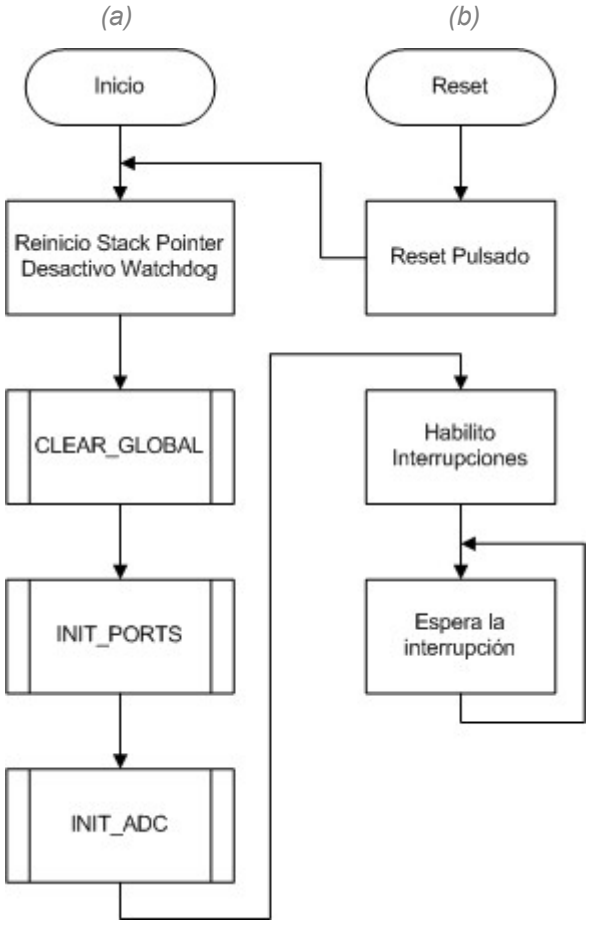


Figura 102. NT0011 – ADC Conv_Cont & Interrupciones. (a) Rutina Principal. Inicializa variables, limpia registros, inicializa puertos y espera que ocurra la conversión. (b) Reinicio del sistema. Al presionar "reset", el sistema reinicia automáticamente.

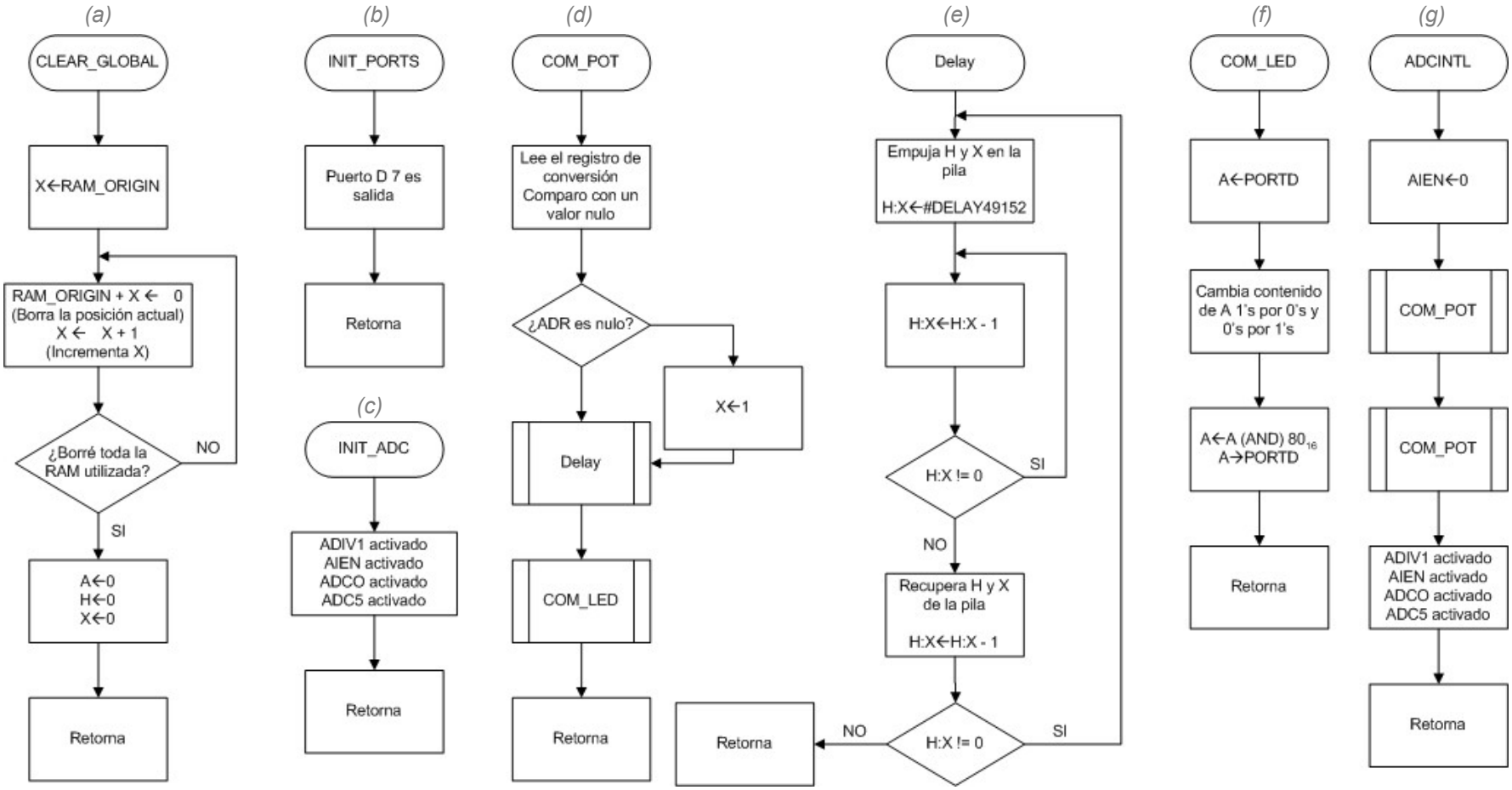


Figura 103. NT0011 – ADC Conv_Cont & Interrupciones - Subrutinas. (a) Clear_Global. Limpia la memoria RAM. (b) Init_Ports. Inicializa el Puerto D7 como salida. (c) Init_ADC. Activa el ADC con interrupciones, conversiones continuas, reloj de ADC de 1.2288 MHz, PTB5 como canal de conversión. (d) Com_Pot. Conmuta el estado del LED dependiendo del valor del potenciómetro. (e) Delay. Retardo programable dependiendo del registro H:X. (f) Com_Led. Conmuta el estado del LED en PTD7. (g) ADCINTL. Petición de Interrupción. Deshabilita interrupciones de ADC y llama a la rutina Com_Pot, luego activa nuevamente las interrupciones y retorna.

1.11.5 Código

```
;*****************************************************************
; ARCHIVO      : NT0011 - ADC Conv_Cont & Interrupciones - 17 03 04.asm
; PROPÓSITO    : Varía la rata de parpadeo de un LED con el potenciómetro.
;                 - Ver el resultado de captura del voltaje
;                reflejado en el LED de la tarjeta.
;                - Se configura el PTB5 para adquirir el valor analógico
;                a digital para luego reflejarlo variando la frecuencia
;                de parpadeo de un LED.
;
;                P.D.:  no hay que hacer ninguna conexión, esta
;                internamente cableado.
; NOTA         :
;                1 - Observar el resultado de la conversión en el registro
;                ADR al entrar en el servicio de interrupción.
;                2 - Observar el resultado de la captura reflejado en la
;                variación de parpadeo del LED de la tarjeta.
; REFERENCIA:
;                Advanced Information of MCU68HC908JK1, JK3, JL3...
;                http://e-www.motorola.com/files/microcontrollers/
;                doc/data_sheet/MC68HC08JL3.pdf
;                Pág. 33 - Registros del ADC
;                Pág. 35 - Vector de Interrupción
;                Págs. 127 @ 136. - Información del ADC
;
;
; LENGUAJE     : IN-LINE ASSEMBLER
; ----------------------------------------------------------------------
; HISTORIAL
;   DD MM YY
;   11  02  03   Creado.
;   31  08  04   Modificado.
;*****************************************************************
$SET    ICS08                   ; ICS08 = 1, Vamos a simular en la pastilla
                                ; la velocidad de simulación es menor en la
                                ; PC.
;$SETNOT  ICS08                 ; ICS08 = 0, Vamos a programar la pastilla
                                ; la aplicación debe correr en tiempo real
$IF ICS08
DELAY49152    equ $0001         ; Constante de Retardo a 4.9152 MHz –
                                ; Simulación
$ELSEIF
DELAY49152    equ $0096         ; Constante de Retardo a 4.9152 MHz –
                                ; Tiempo Real
$ENDIF
```

```
;===============================================
;                  Definiciones del Usuario
;===============================================
COPD            equ 0T                  ; Bit 0 del registro CONFIG1
NULL            equ 0T                  ; Valor Nulo
BIT6            equ 6T                  ; ADSCR, Bit de Interrupción, Bit 6
DUTYTIMEMIN equ $0001                   ; Tiempo mínimo = 1
BIT7            equ 7T                  ; Bit 7 del Registro
BIT7MASK        equ $80                 ; Máscara para el Bit 7 del Puerto
RAM_ORIGIN   equ $0080                  ; Inicio de la memoria RAM
RAM_USED    equ $00FE                   ; Fin de limpieza de la RAM
AIEN            equ %01000000           ; ADSCR, Bit de Interrupción, Bit 6, ON
ADCO            equ %00100000           ; ADSCR, Bit de Conversión Continua, Bit 5
                                        ;ON
CH2             equ %00000100           ; ADSCR, Bit de Canal 0, Bit 2 ON
CH1             equ %00000010           ; ADSCR, Bit de Canal 0, Bit 1 ON
CH0             equ %00000001           ; ADSCR, Bit de Canal 0, Bit 0 ON
ADC7            equ CH2|CH1|CH0         ; ADSCR, PTB7 = ADC
ADIV1           equ %01000000           ; ADIV1, Bit 6 ON

;===============================================
;              Mapa de Memoria del Microcontrolador
;===============================================
;===============================================
;                        Registro de E/S
;===============================================
PORTD           equ $0003               ; Registro del Puerto D
DDRD            equ $0007               ; Registro de Direccionamiento del Puerto D
CONFIG1         equ $001F               ; Vectores de configuración

;===============================================
;                        Registro ADC
;===============================================
ADSCR           equ $003C               ; Dirección, registro de estado y control del
                                        ;ADC
ADR             equ $003D               ; Registro de Datos del ADC
ADICLK          equ $003E               ; Registro de Reloj de Entrada del ADC

;===============================================
;                        Memoria FLASH
;===============================================
FLASH_STARTequ $EC00                    ; Puntero - Mem.FLASH

;===============================================
;                        Vectores de Usuario
;===============================================
ADCINTH         equ $FFDE               ; Conversión Completa (Alto)
RESET_VEC   equ $FFFE                   ; Puntero del RESET
```

```
;=============================================
; OBJETIVO    : Inicio de Codif. del Ensam-
;               blador en Memoria FLASH.
;=============================================
        org FLASH_START             ; Inicio Mem. FLASH

;=============================================
; OBJETIVO    : Configura el ADC para generar
;               una rata de parpadeo gracias
;               al valor registrado del po-
;               tenciómetro.
;=============================================
START
        rsp                         ; Inic.Stack = $00ff
        bset COPD,CONFIG1           ; Desactiva watchdog
        jsr CLEAR_GLOBAL            ; Borra RAM y registros del CPU
                                    ; registros no inicializados.
        jsr INIT_PORTS              ; Subr,Inic. PUERTO
        jsr INIT_ADC                ; Inicializa ADC
        cli                         ; Habilita Interrupciones
ESPERA
        wait                        ; Espera a que se de la primera conversión
        bra ESPERA                  ; Salta al modo de bajo consumo

;=============================================
; CLEAR_GLOBAL      : Borra la Ram utilizada y re-
;                     gistros inherentes
; OBJETIVO    : Borra registros
; ENTRADA     : RAM_ORIGIN, RAM_USED
; SALIDA      : A, H:X y RAM en 0
; REGISTROS
; AFECTADOS   : RAM, H:X, A
; EJEMPLO     :
; Entrada     : RAM_ORIGIN = 0080
;               RAM_USED = 0090
;               X = 7
; Salida      : 0080 @ 0087 = 0
;=============================================
CLEAR_GLOBAL                        ; Borra registros a usar
        ldx #RAM_ORIGIN             ; Carga con el origen
FILL_EMPTY
        clr ,x                      ; rellena con "0" la posición actual
        incx                        ; incrementa puntero de RAM
        cphx #RAM_USED              ; Compara hasta el final deseado
        bne FILL_EMPTY              ; Si no concuerda entonces sigue limpiando
        clra                        ; Borra A
        clrh                        ; Borra H
        clrx                        ; Borra X
        rts                         ; retorna
```

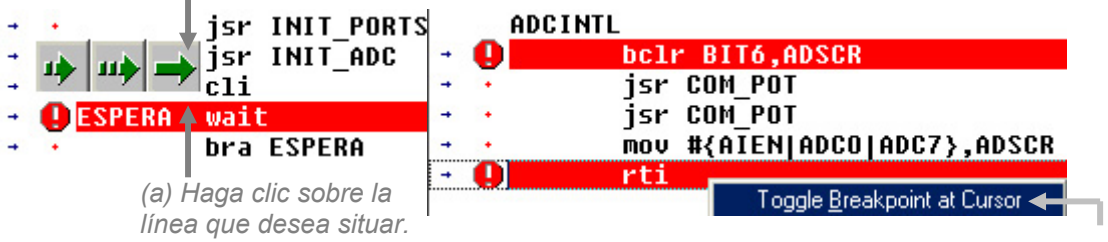
```
;===========================================
; INIT_PORTS  : Inicializa variables y regis
;               tros.
; OBJETIVO    : Inicializa los registros de
;               direccionamiento.
;               PORTD7 = OUTPUT
; ENTRADA     : Ninguna
; SALIDA      : Ninguna
; REGISTROS
; AFECTADOS   : DDRD
;===========================================
INIT_PORTS
        bset BIT7,DDRD              ; Fija PD7 = Salida
        rts                         ; retorna

;===========================================
; INIT_ADC    : Inicializa el ADC
; OBJETIVO    : Conversión Continua e
;               Interrupciones, canal 7
; ENTRADA     : Ninguna
; SALIDA      : Ninguna
; REGISTROS
; AFECTADOS   : ADSCR, ADICLK
;===========================================
INIT_ADC
        mov #ADIV1,ADICLK           ; Reloj del ADC = XTAL/2^(ADIV+2)
        mov #{AIEN|ADCO|ADC7},ADSCR ; Interrupciones habilitadas,
                                    ; Conversión Contínua en PTB7
        rts                         ; retorna

;===========================================
; COM_POT     : Regula la rata de parpadeo
;               de un LED variando el valor
;               del potenciómetro de la tar-
;               jeta.
; OBJETIVO    : Al variar el potenciómetro,
;               varía el tiempo de conmutado
;               del LED
; ENTRADA     : Ninguna
; SALIDA      : Ninguna
; REGISTROS
; AFECTADOS   : X
;===========================================
COM_POT
        ldx ADR                     ; Leo el valor del convertidor
        cpx #NULL                   ; comparo con 0 (la rutina delay retarda de
                                    ; 1 ms a 65.535 seg y no empieza de 0)
        bne DLY0                    ; si no es cero, retardo el tiempo en ms
        ldx #DUTYTIMEMIN            ; Cargo con el mínimo de tiempo: 1 ms
DLY0
        jsr Delay                   ; Genero el retardo
        jsr COM_LED                 ; conmuta el LED del puerto D7
        rts                         ; retorna
```

```
;=========================================
; DELAY       : Genera un retardo de tiempo
; OBJETIVO    : Retardo de tiempo, base 1ms
; ENTRADA     : H:X = Retardo en ms
; SALIDA      : H:X = 0
; REGISTROS
; AFECTADOS   : H:X
; USO         :
;               MIN = H:X = 1T
;               MÁX = H:X = 65535T
;               ldhx #500
;               jsr Delay ; retarda 0.5 seg
;=========================================
Delay
        pshx                    ; [2] Salva X en la pila
        pshh                    ; [2] Salva H en la pila
        ldhx #DELAY49152        ; [3] Carga constante de bucle fino
Delay0
        aix #-1                 ; [2] Decrementa H:X en 1
        cphx #0                 ; [3] LLegó a cero (0)
        bne Delay0              ; [3] Si no es igual, salta a Delay0
        pulh                    ; [2] Si es igual, recupera H de la pila
        pulx                    ; [2] Recupera X de la pila
        aix #-1                 ; [2] Decrementa H:X en 1
        cphx #0                 ; [3] LLegó a cero (0)
        bne Delay               ; [3] Si no es igual, salta a Delay
        rts                     ; [4] retorna

;=========================================
; COM_LED     : Apaga o prende el LED depen-
;               diendo del estado anterior.
; OBJETIVO    : Si el led del puerto D, bit 7
;               estuvo apagado, lo enciende ó
;               visceversa.
; ENTRADA     : Ninguna
; SALIDA      : ACCA = Estado del LED PTD7
; REGISTROS
; AFECTADOS   : ACCA, PORTD
;=========================================
COM_LED
        lda PORTD               ; Carga ACCA con el contenido del Puerto D
        coma                    ; Cambia 1's por 0's y 0's por 1's
        and #BIT7MASK           ; Enmascara el bit 7
        sta PORTD               ; ACCA lo almacena en el puerto D
        rts                     ; retorna
```

```
;===============================================
; ADCINTL      : Interrupción del ADC
; OBJETIVO     : Leer el registro, conmutar el
;                potenciómetro y variar la ra-
;                ta de parpadeo del LED
; ENTRADA      : Ninguna
; SALIDA       : Ninguna
; REGISTROS
; AFECTADOS    : Ninguno
;===============================================
ADCINTL
        bclr BIT6,ADSCR                 ; Deshabilito interrupciones del ADC
        jsr COM_POT                     ; Conmuta el LED a una determinada
                                        ; frecuencia ajustada por el
                                        ; POTENCIÓMETRO
        jsr COM_POT                     ; Conmuta el LED a una determinada
                                        ; frecuencia ajustada por el
                                        ; POTENCIÓMETRO
        mov #{AIEN|ADCO|ADC7},ADSCR     ; Interrupciones habilitadas,
                                        ; Conversión Contínua en PTB5
        rti                             ; retorna de la interrupción

;===============================================
; OBJETIVO     : Inicializa el Vector de Reset
;                Arranque del programa en la
;                memoria Flash y conversión
;                completa del ADC.
;===============================================
;===== Vector de Conversión Completa ADC =======
        org ADCINTH             ; Puntero Vec - ADC
        dw ADCINTL              ; Conversión completa
;===== Vector de Reinicio de Sistema =======
        org RESET_VEC           ; Puntero Vec - RESET
        dw START                ; al darse reset salta a Start
```

Listado 9. NT0011 – ADC – Conv_Cont & Interrupciones. El programa utiliza la interrupción del ADC solamente para leer el registro y retornar; mientras que en el código principal se preocupa de cambiar la rata de parpadeo del LED utilizando el potenciómetro de la tarjeta que permite realizar pruebas experimentales.

1.11.6 Simulador



Nota: Para borrar todos los "breakpoints" escriba en la línea de comandos NOBR.

Figura 104. Uso de "Breakpoints". Utilice los "breakpoints" para depurar un programa o para verificar su funcionamiento. Use los botones de "paso a paso", "multipaso" y "correr" (ver atributo c de esta figura) para evaluar el código, navegue por el mismo.

(a) Inicie WinIDE.

(b) Cargue el archivo NT0011 - ADC Conv_Cont & Interrupciones - 17 03 04.asm.

(c) Compile.

(d) Entre al simulador.

(e) Añada tres (3) "breakpoints" según los pasos (e) y (f).

(e) Presione el botón izquierdo del "mouse" en la línea que desea ubicar un "breakpoint" y luego el botón derecho para ver las opciones.

(f) Seleccione situar un "breakpoint".

(g) Si el breakpoint en "wait" (ver figura 104(a)) causa muchos problemas en simular, remuévalo.

(g) Corra su programa y observe que sucede si varía el potenciómetro.

(h) Si desea borrar todos los "breakpoints", escriba en la línea de comandos la instrucción NOBR.

(i) Corra su programa nuevamente variando el potenciómetro.

(j) Si desea "quemar" su pastilla, revisar la NT0009, Sección 1.9.5.

Nota: Recuerde mover su compilación condicional si desea su programa en tiempo real:

```
;     $SET ICS08
$SETNOT ICS08
```

1.11.7 Conclusión

La conversión continua es un método rápido, ayudado de las interrupciones, para generar lecturas frecuentes del convertidor sin pérdidas de tiempo y aprovechar el uso del CPU para realizar no solo lecturas a un tiempo específico, sino, acciones de control específicas en la rutina principal. Si se desea escribir códigos optimizados en tiempo, utilizar interrupciones es la mejor opción.

Al finalizar la nota, se abarcó el trato de las interrupciones; definir su uso y declaración para el código; luego se mezcla la conversión continua con la interrupción para generar un código que lea en tiempo correcto el convertidor. Por otro lado, el uso de "breakpoints" fue implementado como herramienta de depuración de código, que ayudará más adelante no solo a verificar el funcionamiento de su programa, sino, a encontrar errores de programación poco visibles a la lógica común.

1.11.8 Referencias

1.11.8.1 Interrupciones, conflictos y otras historias
(a) http://www.duiops.net/windows/articulos/irqs.htm

1.11.8.2 Información Avanzada sobre el Microcontrolador
(a) http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC08JL3.pdf

Pág. 33 - Registros del ADC
Pág. 35 - Vector de Interrupción
Pág. 79 - Máscaras
Págs. 127 – 136. - Información del ADC

1.11.8.3 Manual de Referencia del CPU
(a) http://www.freescale.com/files/microcontrollers/doc/ref_manual/CPU08RM.pdf

Pág. 188, Instrucción WAIT.

1.11.8.4 Página "web" sobre esta Nota Técnica
(a) http://www.geocities.com/issaiass/

1.11.9 Problemas Propuestos

1.11.9.1 Realice un programa que lea la variación de dos potenciómetros.

1.11.9.2 Del programa anterior, haga que el mismo diga por medio de leds, cual es mayor al otro.