NT0009

1.9 GENERACIÓN DE SUBRUTINA DE RETARDO DE "SOFTWARE"

CREACIÓN DE LA RUTINA UTILITARIA "DELAY"

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

1.9.1 Introducción

En esta sección se implementará una de las primeras rutinas utilitarias básicas como lo es la generación de retardos. La mayoría de las aplicaciones se diseñan para el usuario, este requiere ver en tiempo real las componentes de control y como el microcontrolador es muy rápido (hablando con respecto al ojo humano) se necesita esta rutina básica para generar esperas y poder visualizar o ver el control que realiza el microcontrolador al elemento controlado. Entre algunos de los puntos importantes a discutir se encuentran:

- El ciclo de instrucción: se traduce en tiempo y depende del cristal, los ciclos de máquina (ciclos de instrucción) son la base principal de la rutina de generación de retardos. Se explica también el uso de subrutinas y porque es importante crearlas.

- Programación de la Memoria Flash: Este es el primer programa que vincula programar un microcontrolador para una aplicación de usuario.

Nota: Una desventaja de las subrutinas de retardo de "software" es que el microcontrolador pierde tiempo haciendo "nada", para un mejor desempeño o control se recomienda el uso del temporizador, pero para fines prácticos muchas veces es necesario tener a mano una rutina básica como la de retardo de "software".

De ahora en adelante cada vez que se utilice una nueva instrucción se referirá al manual del CPU en la página de Motorola, referencia 1.9.7.1, con el fin de interpretar la hoja de datos.

1.9.2 El Ciclo de Instrucción y Retardos

Refiérase en esta sección a las figuras 87 y 88.

1.9.2.1 Tiempos en el CPU

El CPU utiliza un reloj de cuatro (4) fases T1, T2, T3 y T4. El ciclo de instrucción es la base de tiempo en la que se extiende cada instrucción, en este caso, si el microcontrolador posee un cristal, el ciclo de instrucción será de un cuarto (1/4) de la frecuencia del cristal.

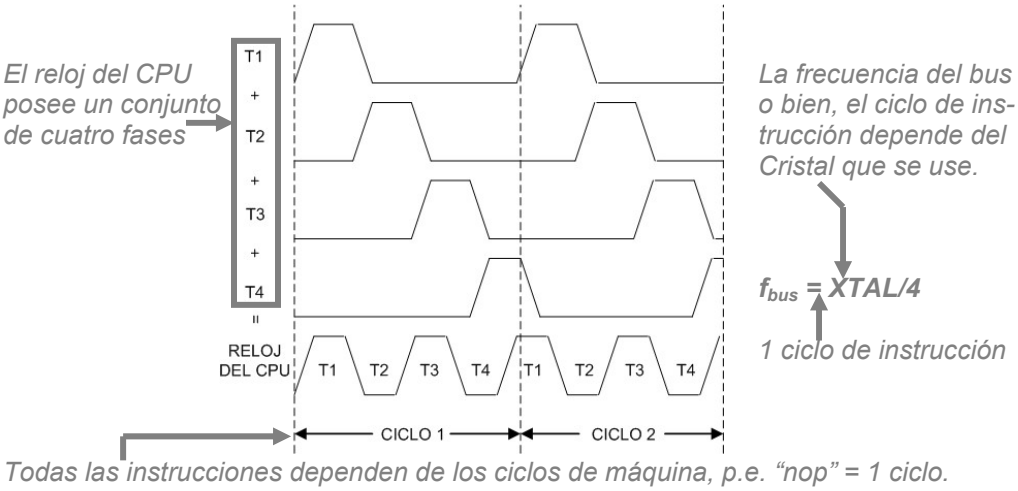


Figura 87. Ciclo de Instrucción. La frecuencia del bus es la cuarta parte del valor del cristal, a esto se le llama un ciclo de instrucción y la duración de este ciclo de instrucción es $1/f_{bus}$.

Ejemplo: Para un cristal de 32MHz; calcular: su frecuencia de bus, el tiempo de un (1) ciclo de instrucción y el tiempo de 8000 ciclos de instrucción.

f_{bus} = XTAL/4 = 32 MHz/4 = 8 MHz [Frecuencia interna máxima del microcontrolador]
$T_{bus} = 1/_{fbus} = 1/(8MHz) = 125$ ns [tiempo de un ciclo de instrucción]
$T_{8000ciclos}$ = Ciclos $\times$ T_{bus} = 8000 $\times$ 125 ns = 1 ms [Tiempo de “x” ciclos de instrucción]

De todo lo anterior se puede concluir que:

$$(4/XTAL) \times Ciclos = t_{base} \qquad \text{[Ecuación General]}$$

Ecuación 1. Ecuación de Cálculo de Tiempos

XTAL = frecuencia del cristal en Hz;
Ciclos = conjunto de ciclos de instrucción a ejecutarse;
t_{base} = tiempo que se presume esperar en segundos.

1.9.2.2 La Función de Una Subrutina

Una subrutina es una secuencia de instrucciones que se utiliza más de una vez en el programa. Cuando el CPU encuentra un llamado a subrutina, este desvía la atención del flujo del programa principal hacia otra parte del código donde ejecuta el contenido de esta subrutina; finalizada la ejecución de dicha subrutina, esta retorna en la siguiente instrucción del código donde fue invocada la subrutina. Es importante el uso de subrutinas pues acelera el proceso de desarrollo de una aplicación, es decir podemos tener bloques de código que controlan dispositivos y de estos solo hacemos un llamado, ejemplo, si se quisiera escribir en una pantalla LCD tendríamos una subrutina LCDWrite que haría esa función.

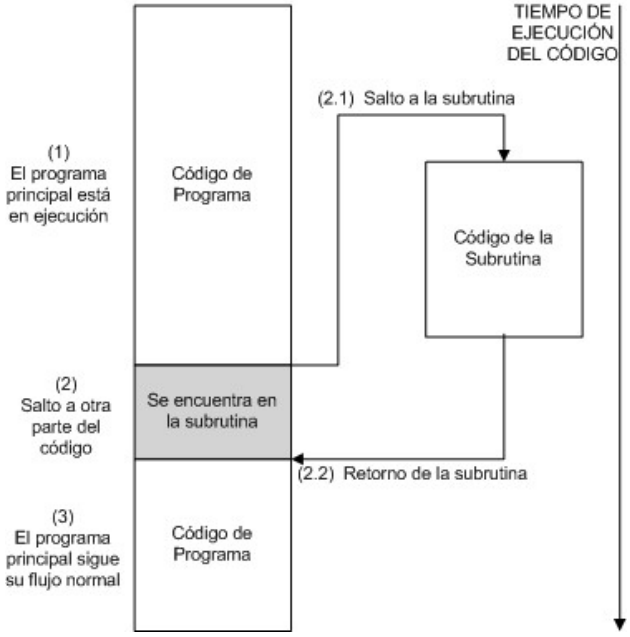


Figura 88. Procedimiento de Ejecución de Una Subrutina. Cuando el programa principal encuentra un llamado a subrutina, este interrumpe el código principal para saltar a otra parte del programa y luego retornar para ejecutar la siguiente instrucción del punto donde se hizo el salto a la subrutina.

Ciclos de Máquina *Código de Subrutina*

```
[02]          Delay     pshx
[02]                    pshh
[03]                    ldhx #DELAY49152
[02]          Delay0    aix #-1
[03]                    cphx #0
[03]                    bne Delay0
[02]                    pulh
[02]                    pulx
[02]                    aix #-1
[03]                    cphx #0
[03]                    bne Delay
[04]                    rts
```

Listado 4. Código de la Subrutina de Retardo. Los ciclos de máquina de cada instrucción se encuentran en el manual del CPU, ver la referencia 1.9.7.1. Esta es una subrutina programable en la cual su base está en milisegundos, permite un máximo de 65.535 segundos de retardo.

1.9.2.3 Cálculo del Tiempo de la Subrutina de Retardo – "Delay"

Refiérase en este punto al listado 4 y 5.

1.9.2.3.1 Llamado de la Subrutina de Retardo

La subrutina *"Delay"* es capaz de generar retardos desde 1 ms hasta 65.535 seg., pero esto no asegura de que sean exactamente la cantidad de tiempo precisa, este es solo un estimado. Su llamado se realiza cargando el tiempo en el registro HX y luego invocándola:

```
Ciclos de Máquina   Código del Llamado
[03]                ldhx #1        ; a retardar 1 ms
[05]                jsr Delay      ; ejecuta el retardo
```

Listado 5. Llamado de la Subrutina de Retardo. El registro H:X almacena el valor de la cantidad a retardar y luego es invocada por la instrucción JSR seguido del nombre de la subrutina.

1.9.2.3.2 Cálculo del Retardo

Se inicia por el llamado (listado 5) y luego por el cuerpo de la subrutina (listado 4) sumando uno y cada uno de los ciclos de máquina.

$$3 + 5 + [2 + 2 + 3 + (2 + 3 + 3) \times DELAY49152 + 2 + 2 + 2 + 3 + 3] \times H{:}X + 4$$
$$12 + (19 + 8 \times DELAY49152) \times H{:}X$$

Si la base de tiempo es 1 ms (t_{base}) entonces H:X es conocido (H:X = 0001)

$$Ciclos = 31 + 8 \times DELAY49152$$

Según la ecuación 1 en el ejemplo de la sección 1.9.2.1 se puede concluir que:

$$(4/4915200) \times (31 + 8 \times DELAY49152) = 0.001$$

$$DELAY49152 = 149.725 \sim= 150_{10} = 96_{16}$$

La siguiente tabla muestra el valor de DELAY49152 para algunos cristales.

Tabla 45. Cristales Comerciales y Valor de Variable de Retardo

Cristal en MHz	DELAY49152
4	0079_{16}
4.9152	0096_{16}
32	$03E4_{16}$

Nota: A diferentes frecuencias de 4.9152 MHz no se conectará *la tarjeta de desarrollo* a los modos de simulación o programación, la tarjeta es fija en esta frecuencia.

1.9.3 Diagrama de Flujo

El siguiente programa verifica el estado del jumper PTB5 (serigrafiado en la tarjeta), si está levantado parpadea más rápido; si está en su posición original parpadea más lento. Se utiliza una rutina básica de retardo programable cuya base de tiempo es de 1 ms.

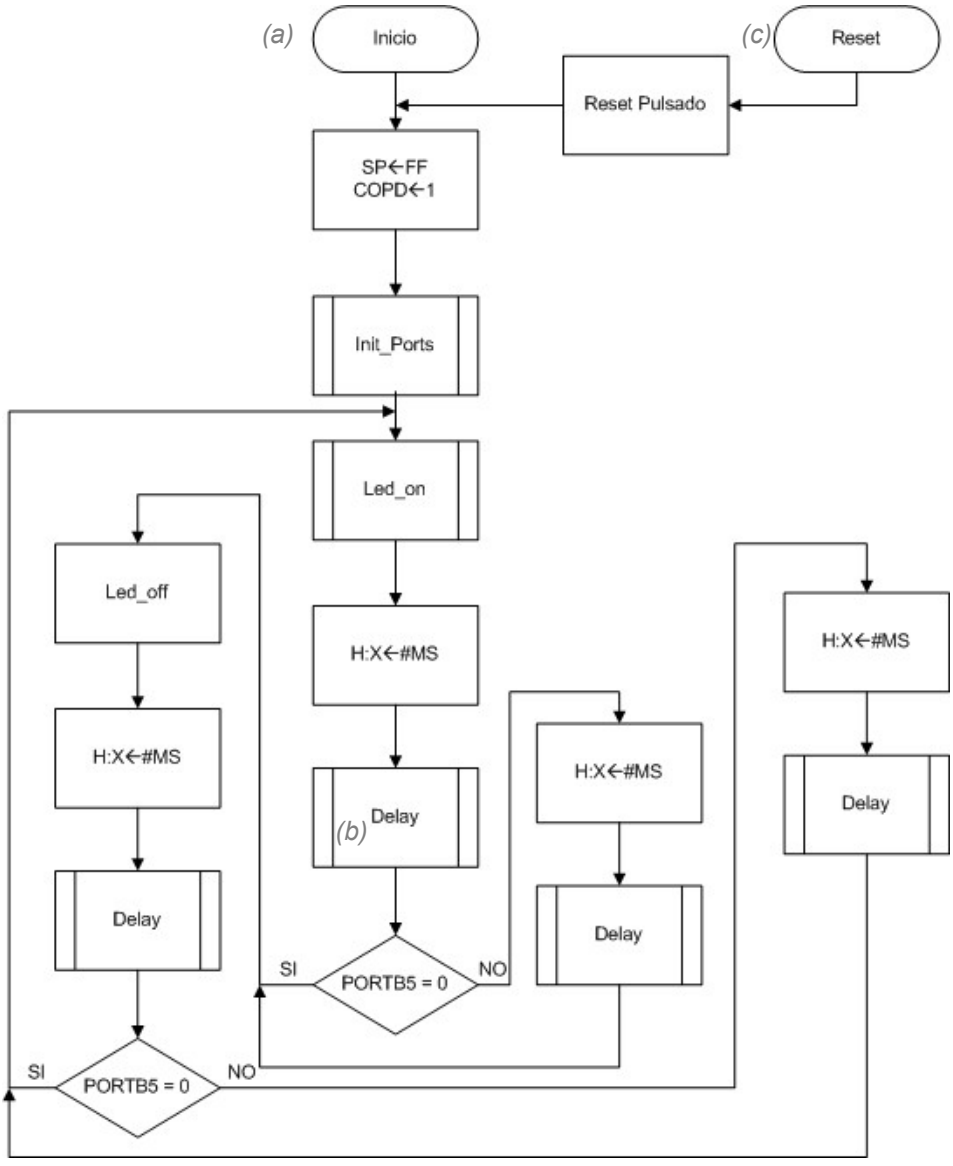


Figura 89. Diagrama de Flujo de NT0009 - Retardos. (a) Programa Principal. Prende o apaga un LED dependiendo del estado del jumper PTB5. (b) Subrutina de retardo. Demora de "software". (c) "Reset". Reinicio del sistema.

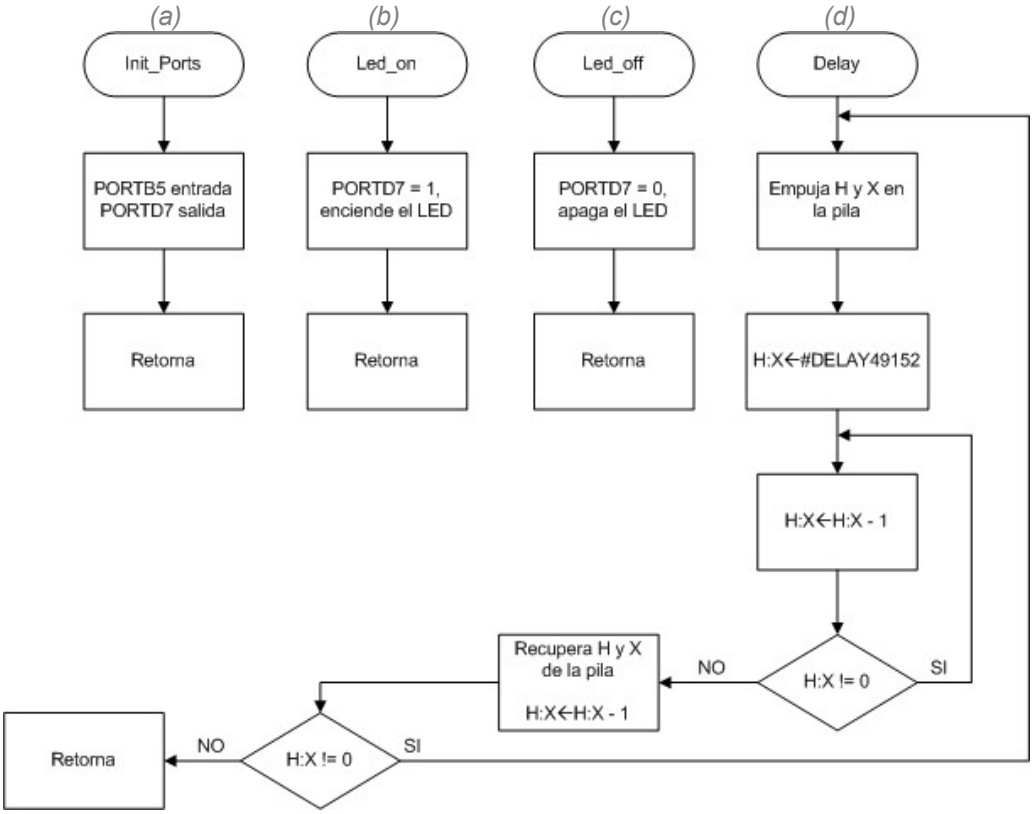


Figura 90. Diagrama de Flujo de NT0009 - Subrutinas. (a) Inicialización de los puertos. Puertos B5 y D7 como entrada y salida respectivamente. (b) Encendido del LED de la tarjeta. Puerto D7 en estado alto (1). (c) Apagado del LED de la tarjeta. Puerto D7 en estado bajo (0). (d) Retardo de “software”. Retardo programable por el registro H:X de 1ms de retardo.

1.9.4 Código

```
;*****************************************************************
; ARCHIVO      :  NT0009 - Retardos - 25 02 04.asm
; PROPÓSITO  : Verificar el Jumper del Pin PB5
;              Probar el retardo programable de
;              base de tiempo de 1 ms
;              Parámetros de Prueba:
;                     El LED    - Puerto D - Pin D7
;                     Puerto D  - DDRD = 1000 0000 = $80 ( D7 = Salida )
;                     El JUMPER - Puerto B - Pin B5
;                     Puerto B  - DDRB - 1101 1111 = $DF ( B5 = Entrada )
;
;              Verificación Estática del Jumper con el Simulador:
;              1. DDRB DF
;              2. Colocar y Remover el Jumper
;              3. Observar el "Memory" & "Variables" Window
;
;              Verificación Dinámica del Jumper - Hardware & Software
;              1. Usando esta versión del programa DEMO01
;              2. Colocar y Remover el Jumper
;              3. Observar la Variación de Velocidad de ON/OFF del LED
;
; LENGUAJE    : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   10   01   00   Creado.
;   27   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                 ; Si vamos a simular (ICS08 = 1), entonces
MS              equ $000A       ; Retardo en milisegundos para simular
$ELSEIF                         ; De lo contrario (ICS08 = 0), entonces
MS              equ $01F4       ; Retarda 500 ms, 01F4 = 500.
$ENDIF                          ; Fin de la compilación condicional

;*****************************************
;          Mapa de Memoria del Microcontrolador
;*****************************************
;*****************************************
;                    Registro de E/S
;*****************************************
PORTB           equ $0001       ; Registro del Puerto B
PORTD           equ $0003       ; Registro del Puerto D
DDRB            equ $0005       ; Registro de Direccionamiento del Puerto B
DDRD            equ $0007       ; Registro de Direccionamiento del Puerto D
```

```
;=============================================
;                Registro de Configuraciones
;=============================================
CONFIG1     equ $001F           ; Puntero - Reg. de Configuración

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

;=============================================
;                      Vectores de Usuario
;=============================================
RESET_VEC   equ $FFFE           ; Puntero del RESET


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

;=============================================
; OBJETIVO    : Inicio del programa
;               SP   = STACK POINTER
;               COPD = WATCHDOG
;=============================================
START       rsp                 ; inic.Stack = $00ff
            bset COPD,CONFIG1   ; desactiva watchdog
            jsr Init_Ports      ; Subr, Inic. PUERTO

;=============================================
; PROPÓSITO   : Ciclo interminable, prende y
;               apaga un LED a intervalos de
;               tiempos dada por la subrutina
;               de retardo.
;=============================================
Ciclo       jsr Led_on          ; Subr.ENCENDER LED
            ldhx #MS            ; [3] A retardar 0.5 seg
            jsr Delay           ; [5] Ejecuta el retardo, llamado de la subrutina
            brclr 5,PORTB,APAGAR
                                ; PB5 = 0 ?
            ldhx #MS            ; A retardar 0.5 seg
            jsr Delay           ; PB5 = 1
APAGAR      jsr Led_off         ; Subr. APAGAR LED
            ldhx #MS            ; A retardar 0.5 seg
            jsr Delay           ; Ejecuta el Retardo
            brclr 5,PORTB,ENCENDER
                                ; PB5 = 0 ?
            ldhx #MS            ; A retardar 0.5 seg
            jsr Delay           ; Ejecuta el Retardo
ENCENDER    bra Ciclo           ; Repite el ciclo
```

```
;===========================================
; INIT_PORTS  :  Inicializa variables y regis
;                tros.
; OBJETIVO    :  Inicializa los registros de
;                direccionamiento.
;                PORTB5 = INPUT
;                PORTD7 = OUTPUT
; ENTRADA     :  Ninguna
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  DDRB, DDRD
;===========================================
Init_Ports      bclr 5,DDRB          ; Fija PB5 = Entrada
                bset 7,DDRD          ; Fija PD7 = Salida
                rts                  ; retorna

;===========================================
; LED_ON      :  Enciende el LED del PORTD7
; OBJETIVO    :  PORTD7 = ON
; ENTRADA     :  Ninguna
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  PORTD
;===========================================
Led_on          bset 7,PORTD         ; BIT 7 = 1 = LED = ON
                rts                  ; retorna

;===========================================
; LED_OFF     :  Apaga el LED del PORTD7
; OBJETIVO    :  PORTD7 = OFF
; ENTRADA     :  Ninguna
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  PORTD
;===========================================
Led_off         bclr 7,PORTD         ; BIT 7 = 0 = LED = OFF
                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

;===========================================
; OBJETIVO:  Inicializa el Vector de Reset
;         Arranque del programa en la memo-
;         ria Flash.
;===========================================
;===== Vector de Reinicio de Sistema =======
                org RESET_VEC           ; Puntero Vec - RESET
                dw  START               ; al darse reset salta a Start
```

Listado 6. NT0009 – Retardos. La función principal de la rutina es evaluar el estado del jumper PTB5, si está levantado parpadea al doble de la velocidad de retardo. La subrutina de retardo se programa utilizando el registro H:X, cargando el tiempo equivalente. Se explora el uso de las instrucciones JSR y RTS para llamado y retorno de subrutinas, BCLR y BSET para asignar valores booleanos a bits y finalmente BRCLR y BRSET para probar el estado lógico del bit. Para ver la función de cada nueva instrucción refiérase a la sección 1.9.7.2.

1.9.5 Programado del Microcontrolador – PROG08SZ

(1) Abrir el archivo en WinIDE

(2) Cambiar la configuración como se muestra en el siguiente listado

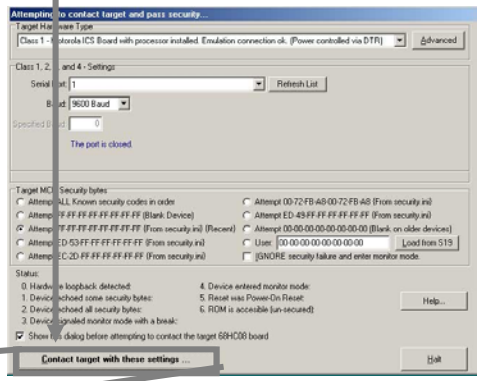
```
;$SET    ICS08          ; ICS08 = 1, Vamos a simular en la pastilla
                        ; la velocidad de simulación es menor en la
                        ; PC.
$SETNOT  ICS08          ; ICS08 = 0, Vamos a programar la pastilla
                        ; la aplicación debe correr en tiempo real
```

(4) Inicie el programador

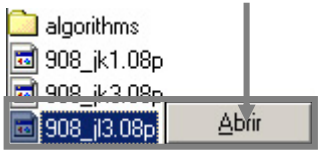
(3) Compilar el programa

(a)

(5) Inicie el programador haciendo clic en "Contact target with this settings"



*(6) Seleccione el algoritmo de su microcontrolador, por ejemplo **908jl3.08p***



(b)

Figura 91. Programando el Microcontrolador (1). (a) Iniciando el programador. Cargue el programa ejemplo, cambie los parámetros de compilación condicional para aplicación en tiempo real; luego compile e inicie el programador. (b) Contactando el programador. En la ventana de contacto presione "Contact target with this settings…" y seguido seleccione el algoritmo que corresponde a su microcontrolador.

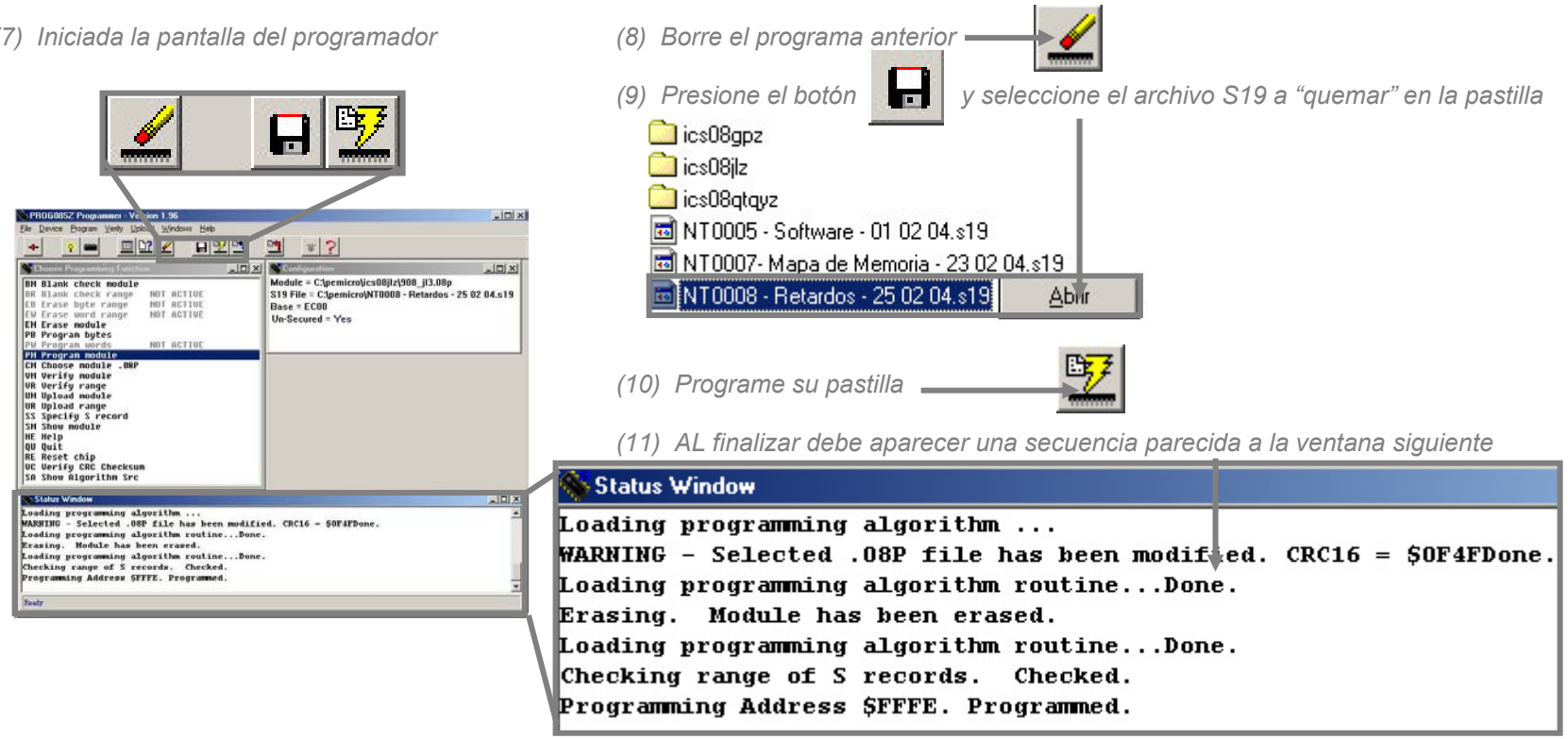


Figura 92. Programando el Microcontrolador (2). Iniciada la pantalla del programador, borre el programa anterior residente en la memoria FLASH; busque el archivo de extensión S19 que desee programar en su microcontrolador para luego programar la pastilla. Finalizada la secuencia debe aparecer una línea de comentarios generados por el programa parecidos a la existente en la ventana de estado de este tutorial.

En esta sección refiérase a las figuras 91 y 92.

(a) Inicie WinIDE.

(b) Abra el archivo en ensamblador: NT0009 – Retardos – 25 02 04.asm.

(c) Cambie el parámetro de la NT0009 según el listado de la figura 91(2).

(d) Compile el archivo.

(e) Inicie el Programador – PROG08SZ.

(f) En la ventana de contacto seleccione “Contact target with this settings…”.

(g) Seleccione el algoritmo correspondiente de su microcontrolador.

(h) Presione [] para borrar el programa anterior de su pastilla.

(i) Presione [] y seleccione el archivo de extensión S19 a programarse.

(j) Presione [] para programar su microcontrolador y espere a ser programada.

(k) Cambie el estado del jumper a aplicación, encienda y reinicie la tarjeta.

(l) Levante el jumper (PTB5) y observe como varía la rata de parpadeo del LED; baje el jumper (PTB5) y observe como varía la rata de parpadeo del LED.

1.9.6 Conclusión

Se implementó una rutina básica de retardo de fácil configuración como lo es “Delay”, la cual permite generar retardos de 1 hasta 65535 milisegundos. Esto facilita la programación de una aplicación sin necesidad de generar tantas rutinas de retardos.

Con el fin de ilustrar el ejemplo de retardo se hizo un análisis superficial del tiempo que toma un ciclo de máquina o ciclo de instrucción. Seguido, una definición de la función de la subrutina, su llamado, retorno y el cálculo de la misma.

Luego con el fin de ilustrar el ejemplo se programó la rutina en la pastilla completando así la primera aplicación básica en tiempo real que utiliza el jumper de la tarjeta para generar dos secuencias de parpadeo de un LED, cada una aproximadamente entre 500 ms y 1000 ms.

1.9.7 Referencias

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

Pág. 33, Tiempo del CPU
Pág. 105, Instrucción BCLR
Pág. 128, Instrucción BRCLR
Pág. 130, Instrucción BRSET
Pág. 131, Instrucción BSET
Pág. 132, Instrucción BSR
Pág. 149, Instrucción JSR
Pág. 171, Instrucción RTS

1.9.7.2 “HC05 Family - Understanding Small Microcontrollers”
(a) http://www.freescale.com/files/microcontrollers/doc/ref_manual/M68HC05TB.pdf

Pág. 76, Llamado y retorno de una subrutina
Pág. 141, Programa de subrutina de retardo
Pág. 314, Definición de una subrutina

1.9.7.3 Página “web” sobre esta Nota Técnica
(a) http://www.geocities.com/issaiass/

1.9.8 Problemas Propuestos

Para los siguientes problemas refiérase a la sección 1.9.7.1

1.9.8.1 Utilice la instrucción BSR para generar subrutinas en un programa que prenda y apague el LED a intervalos de tiempo de 1000 ms y 750 ms.

1.9.8.2 Utilice la instrucción BRSET, si el jumper está levantado haga que parpadee por 3000 ms, sino, preserva su último estado (ya sea encendido o apagado).

1.9.8.3 Realice una rutina que cuente cuantas veces ha levantado y bajado el jumper PTB5. Nota: Reserve dos variables en RAM una para “levantado” y otra para “bajado” del jumper.