Cika Electrónica

Nota de Aplicación: CAN-071
Título: **Software timers en Holtek**
Autor: Sergio R. Caprile, Senior Engineer

| Revisiones | Fecha | Comentarios |
|---|---|---|
| 0 | 23/02/07 | port de CAN-069 |
| 1 | 10/08/07 | optimización, reducción de uso de RAM |
| | | |

Los micros de la familia HT48 de Holtek incorporan entre uno y tres hardware timers. Particularmente en los que disponen de uno solo, algunos tipos de aplicación requieren de un sistema de software timers para controlar eventos en tiempos del orden de las decenas de milisegundos hasta las decenas de segundos. Presentamos aquí un sencillo y eficiente esquema de software timers de 8-bits, controlados por un hardware timer del micro.

La idea es muy simple: el timer de 8-bits (el único para dispositivos menores a HT48E50) interrumpe cada 10ms; su rutina de interrupción se encarga de llevar la cuenta para contar segundos y decrementar los timers en uso:

```
T0HND:          mov softstack,A                     ; salva A
                mov A,STATUS
                mov softstack[1],A                  ; salva STATUS
                mov A,MP0
                mov softstack[2],A                  ; salva MP0

                mov A,OFFSET _CS_TIMERS             ; indirecto
                mov MP0,A
                mov A,NUM_CSTIMERS                  ; para cada timer
t0l1:           sz IAR0                             ; evita si es 0
                dec IAR0                            ; decrementa
                inc MP0                             ; siguiente
                sdz ACC
                jmp t0l1                            ; loop

                dec c1s                             ; decrementa contador de 100
                snz Z
                jmp t0l3                            ; no es 0, chau
                mov A,100                           ; inicializa contador
                mov c1s,A

                mov A,OFFSET _SEC_TIMERS            ; procesa timers en segundos
                mov MP0,A
                mov A,NUM_SECTIMERS
t0l2:           sz IAR0
                dec IAR0
                inc MP0
                sdz ACC
                jmp t0l2

t0l3:           mov A,softstack[2]
                mov MP0,A                           ; recupera MP0
                mov A,softstack[1]
                mov STATUS,A                        ; recupera STATUS
                mov A,softstack                     ; recupera A
                reti
```

Agregamos además una simple rutina de inicialización para configurar el timer y poner todos los software timers a cero

```
NUM_CSTIMERS   equ 5
NUM_SECTIMERS  equ 5

clockrate equ 4000000                       ; clock rate
prescaler      equ    256                   ; prescaler (configurar más adelante !)
TMRCONSTANT    equ    256-(clockrate/(prescaler*100))      ; 10ms
```

```
                public _init_softtimers,_CS_TIMERS,_SEC_TIMERS

stvars  .section 'data'
c1s             db ?
_CS_TIMERS      db NUM_CSTIMERS dup(?)
_SEC_TIMERS     db NUM_SECTIMERS dup(?)
softstack       db 3 dup(?)


stc             .section 'code'

; llamar con interrupciones no habilitadas
_init_softtimers:
                mov A,100                                       ; 100x10ms = 1s
                mov c1s,A
                mov A,OFFSET _CS_TIMERS                         ; indirecto
                mov MP0,A
                mov A,NUM_CSTIMERS                              ; para cada timer
isl1:           clr IAR0                                        ; borra
                inc MP0                                         ; siguiente
                sdz ACC
                jmp isl1                                        ; loop
                mov A,OFFSET _SEC_TIMERS
                mov MP0,A
                mov A,NUM_SECTIMERS                             ; para cada timer
isl2:           clr IAR0
                inc MP0
                sdz ACC
                jmp isl2
                mov A,TMRCONSTANT                               ; divide a 10ms
                mov TMR,A
                mov A,10010111b                                 ; timer, prescaler=256
                mov TMRC,A
                set INTC.2                                      ; habilita int del timer
                ret
```

Finalmente, un sencillo programa de ejemplo para observar cómo pueden utilizarse los software timers desde assembler:

```
                call _init_softtimers
                set INTC.0                                      ; habilita interrupciones

                mov A,5
                mov _SEC_TIMERS[4],A                            ; durante 5 segundos
                set PB.1
l2:             mov A,28                                        ; demora de 280ms
                mov _CS_TIMERS[3],A
l1:             sz _CS_TIMERS[3]
                jmp l1
                mov A,2
                xorm A,PB                                       ; toggle pin
                sz _SEC_TIMERS[4]                               ; pasó el tiempo ?
                jmp l2
                clr PB.1
                set PC.0
```

y por qué no también desde C:

```
        INIT_SOFTTIMERS();
        _intc|=(1<<0);                              // habilita interrupciones

        SEC_TIMERS[4]=5;
        _pb|=1<<1;

        while(SEC_TIMERS[4]){
                CS_TIMERS[3]=28;
                while(CS_TIMERS[3]);
                _pb^=(1<<1);
        }
```

Dado que inicializamos el timer cuando queremos, y éste es decrementado por la rutina de interrupciones, que corre libre, la precisión que tenemos es de una cuenta en defecto, es decir, si seteo un timer en un valor

justo en el instante en el que la rutina de interrupciones se ejecutó, tendré la demora especificada; si lo hago justo en el instante anterior a que la rutina de interrupciones se ejecute, tendré una cuenta menos. Entonces, el tiempo de demora de esta implementación es de $\left(x-\frac{1}{2}\right)\pm\frac{1}{2}$ cuentas. Por ejemplo: 28 en _CS_TIMERS (cuentas de 10ms) es en realidad $275\pm5\,ms$ y 5 en _SEC_TIMERS es $4{,}5\pm0{,}5\,seg$ .

Si se desea mayor precisión se puede emplear una combinación de un timer corto y uno largo: el timer corto puesto en 1 detecta el instante de cuenta, y a continuación se inicia el timer largo.

Esta discusión asume que el cristal es de una frecuencia tal que dividido por el prescaler, para contar 10ms, el valor a incorporar en el registro de control del timer es un entero. Esto no es así con cristales de valores "redondos" (4MHz, por ejemplo), pero sí con cristales "de UART" (3,6864MHz, por ejemplo).

El código fue escrito para HT48E30, y puede utilizarse sin modificaciones en gran cantidad de dispositivos, requiriendo menores modificaciones para micros con mayor cantidad de timers. Por ejemplo, deberemos referirnos a TMR0 y TMR0C en vez de TMR y TMRC en el HT48E50. Para otros casos (y por qué no este también), se recomienda la lectura de la correspondiente hoja de datos.