Índice

1. Introducción

Esta nota es la documentación de unas rutinas de retardo muy simples escritas en assembler para Rabbit.

1.1. Objetivos

El objetivo es realizar una rutina en lenguaje ensamblador para generar retardos del orden de 1us en adelante que puedan ser útiles a desarrolladores que desconozcan el lenguaje, pero que en algún aspecto de su aplicación requieran utilizarlo, puesto que el lenguaje C no les brinda la suficiente precisión.

1.2. Aclaraciones

Este trabajo no intenta brindar nociones del lenguaje utilizado ni de la metodología de implementación dentro del programa de Dynamic C. Para mayores datos, referirse a la documentación provista por el fabricante:

- Dentro del entorno de desarrollo Dynamic C en Help->Instruction Set Reference [Alt+F1]
- En el sitio web de documentación del fabricante:
http://www.rabbit.com/docs/ en la sección *Microprocessors*
-Manual de Dynamic C, Sección 13, “Using Assembly Languge”
http://www.rabbit.com/documentation/docs/manuals/DC/DCUserManual10/Dynamic_C_Users_Manual.htm
Libros de Sergio Caprile, Cika Electrónica:
    - Desarrollo con Procesadores y Módulos Rabbit
    - El camino del Conejo.

2. Las rutinas

Teniendo en cuenta que muchos desarrolladores no manejan el lenguaje ensamblador ni disponen de tiempo y/o ganas para aprenderlo realizamos un par de rutinas en dicho lenguaje para Rabbit.

2.1. Rutina de retado variable

Tiene la simple finalidad de generar retardos entre 3 y 65535 microsegundos, teniendo en cuenta la frecuencia de operación del microprocesador y con independencia del modelo que estemos usando ya que obtenemos la unidad de tiempo de la variable del sistema `freq_divider`:

```
long  CPUfreq;

#if CC_VER < 0xA50                          // for versions < 10.50
   CPUfreq = (long)freq_divider * 6144L;
#else
   CPUfreq = get_cpu_frequency();           // more accurate calculation
#endif

loopsPer_us = (unsigned int) (CPUfreq /(LOOP_CYCLES*10000L)) ;

delay_us(1);

delay_fixed_us();
```

Así generamos el valor de `loopsPer_us` que le indica a la rutina de delay cuantos loops son necesarios para generar un retardo de aproximadamente 1us:

```
// ld hl, 0x0000            ;6 \ previous calling time:
// push hl                  ;10 |1.27 us aprox.
// call delay_us            ;12/
unsigned int loopsPer_us;
void delay_us(int us)
{
#asm
   ld hl, (sp + 0x02)       ;9 \ get us \
   ex de, hl                ;11/          | setup
   ld hl, (loopsPer_us)     ;11           | time = 61
   ld b, h                  ;2            |
   ld c, l                  ;2            | clk = 22Mhz
   mul                      ;12           | => 2.72 us
   ld h, b                  ;2\           | aprox.
   ld l, c                  ;2 |Return    |
   bool hl                  ;2 |if zero   |
   ret z                    ;8/          /
loopa:
   ld h, b                  ;2 \
   ld l, c                  ;2 | loop time
   dec hl                   ;2 | LOOP_CYCLES = 19
   ld b, h                  ;2 | clk = 22Mhz
   ld c, l                  ;2 | => 0.86 us
   bool hl                  ;2 | aprox.
   jp nz, loopa             ;7 /
#endasm
}
#define LOOP_CYCLES 19
```

Como puede verse, hay unos tiempos adicionales que es necesario considerar, o al menos saber que existen, estos son, el tiempo de preparación de los registros para hacer el loop (3us aprox.) y el tiempo de llamada y retorno (aprox. 1 us. cada uno).

2.1. Rutina de retado fija

La rutina restante es de tiempo fijo, de alrededor de 1us y puede servir de ejemplo para hacer retardos en ese orden de tiempo. En este caso, se tuvo en cuenta un clock de 22Mhz para calcular el tiempo de retardo, así que esta rutina no sería directamente portable sin considerar el clock del sistema al querer utilizarla con un microprocesador que funcione con un clock diferente.

```
//Fixed 1us aprox.
void delay_fixed_us()
{
#asm debug
   ld b,3        ; 4
again:           ;
   nop           ; 2
   djnz again    ; 5
#endasm
}
```