Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°6

Timers

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1°/2020								
Turno de las clases prácticas			Miércoles								
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos								
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci								
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Santiago	López	100566									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivo

Hacer uso de los timers del micro para manejar la frecuencia de oscilación de un LED de acuerdo al valor de entrada.

2. Descripción

Se reciben dos señales digitales en los pines PD0 y PD1. En base al valor de ambos bits se determina el prescaler para dividir la frecuencia del clock utilizado, y así variar el tiempo en el que se produce un overflow en el contador del timer del micro. El overflow provoca una interrupción durante la cual se cambia el estado de un LED, demostrando así la frecuencia a que trabaja el timer.

Los estados a leer son el 0b00, 0b01, 0b10 y 0b11, para los cuales corresponden los estados `encendido fijo` y parpadeo con `prescaler clock / 64`, `prescaler clock / 256` y `prescaler clock / 1024`, respectivamente.

3. Diagrama de conexiones en bloque

Las conexiones siguieron el esquema de la Figura 1.

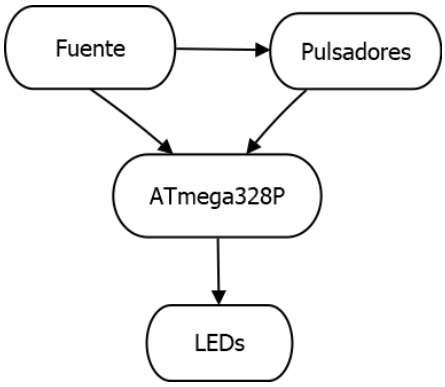

Figura 1: Diagrama de conexiones en bloque.

4. Esquemático

La Figura 2 muestra las conexiones eléctricas efectuadas en el práctico. Los componentes utilizados se encuentran en la sección siguiente. En caso de querer utilizarse la resistencia de *pull-up* interna del micro, se conectaría cada puslador entre el pin de entrada y tierra, configurando el progama para que interprete los 0V como '1' lógico. De esta forma se ahorra el uso de los resistores externos de 10kΩ.

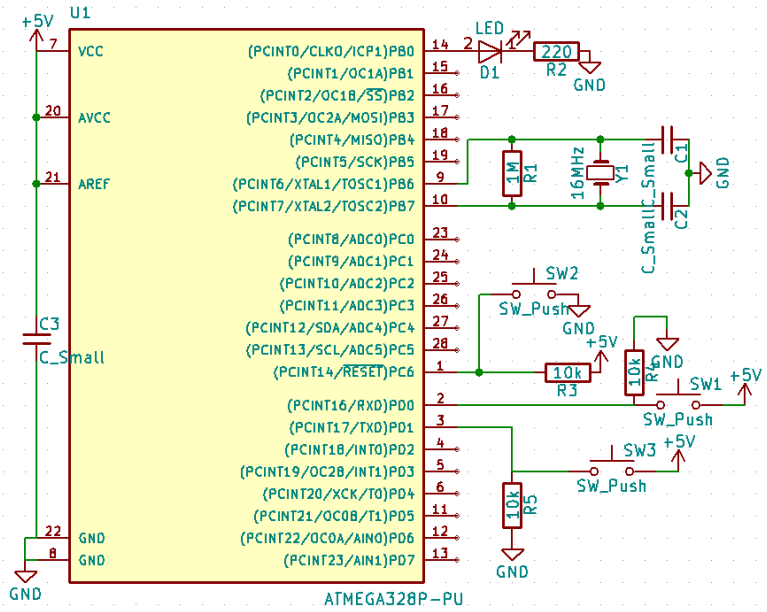

Figura 2: Circuito esquemático.

5. Listado de componentes

Los componentes utilizados fueron los listados a continuación:

- Placa de desarrollo Arduino UNO
- Resistores de 220Ω y 10kΩ
- LED de color rojo
- Pulsadores
- Protoboard
- Cables unipolares

6. Diagrama de flujo

En la Figura 3 se presentan los pasos a seguir por el programa. La primera rutina lleva los setups de los puertos y del stack del micro, y finalmente un loop de llamado a la rutina de lectura y de seteado del prescale. La segunda rutina determina qué prescaler setear en el registro TCCR1B en base a la tabla de estados y el valor de entrada leído. La tercer rutina representa la interrupción efectuada al producirse un overflow en el timer 1. Por último, la cuarta rutina sigue los pasos efectuados al leer un estado, en donde se verifica si el estado es nuevo o no, y luego se hace un chequeo en el caso de que haya un rebote mecánico.

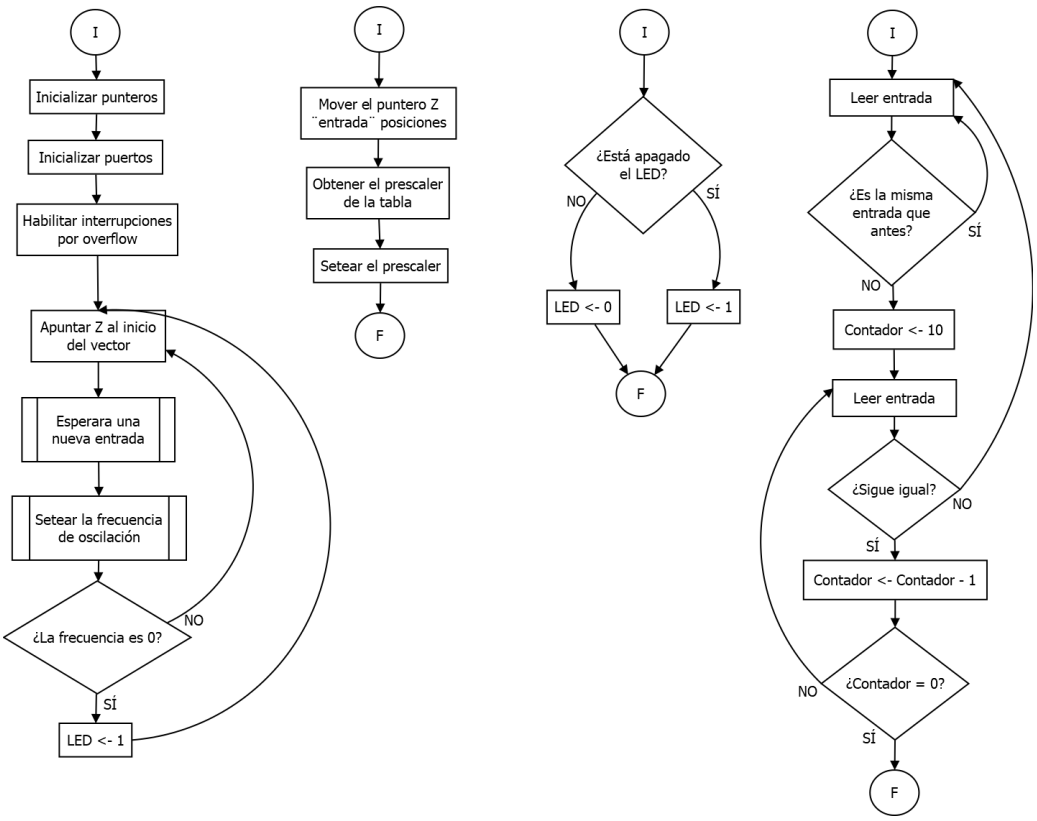

Figura 3: Diagrama de flujo.

7. Código fuente

El programa consiste en el setup de los registros y en buscar el código de prescale para setear la frecuencia de oscilación. Mientras tanto, cada vez que el timer tiene un overflow se produce una interrupción en donde se alterna el estado del LED.

Se implementó una tabla para hallar el estado adecuado para cada prescale con el fin de generalizar el algoritmo ante el caso de tener que añadir o quitar frecuencias de oscilación.

```
   1: .include "m328pdef.inc"
   2:
   3: .dseg
   4:
   5: .def conf = r16
   6: .def freq = r17
   7: .def aux = r18
   8: .def aux1 = r19
   9: .def last_input = r20
  10: .def count = r21
  11: .equ input = pind
  12: .equ led_port = portb
  13: .equ led = 0
  14: .equ len = 4
  15:
  16:
  17: .macro init_sp
  18:     ldi conf, low(RAMEND)
  19:     out spl, conf
  20:     ldi conf, high(RAMEND)
  21:     out sph, conf
  22: .endmacro
  23:
  24:
  25: .cseg
  26: .org 0x0000
  27:     jmp main
  28: .org 0x001A
  29:     jmp timer_isr
  30:
  31:
  32:
  33: .org INT_VECTORS_SIZE
  34:
  35: main:
  36:     init_sp
  37:     call setup_ports
  38:
  39:     call interrupt_enable
  40:     sbi led_port, led
  41:
  42: here:
  43:     call init_zp        ; apunto z al vector de prescales
  44:     call wait4input
  45:     call set_frequency
  46:     cpi freq, 0x00
  47:     brne here
  48:     sbi led_port, led   ; si no setee prescale dejo me aseguro que quede prendido el
led
  49:     jmp here
  50:
  51:
  52: wait4input:
  53:     in aux, input
  54:     cp aux, last_input  ; si es la misma que antes no hago nada
  55:     breq wait4input
  56:     ldi count, 10       ; leo 10 veces la entrada a ver si se mantiene estatica
  57: check_read:
  58:     dec count
  59:     in aux1, input
  60:     cp aux1, aux
  61:     brne wait4input     ; si varia la entrada, vuelvo a leer
  62:     cpi count, 0x00
  63:     brne check_read
  64:     mov last_input, aux ; actualizo el ultimo estado
  65: ret
  66:
  67: set_frequency:
  68:     clr aux
```

```
 69:     add zl, last_input
 70:     adc zh, aux
 71:     lpm freq, z     ; cargo el valor del vector
 72:     sts tccr1b, freq
 73: ret
 74:
 75: interrupt_enable:
 76:     ldi conf, 0x01
 77:     sts timsk1, conf ; interrupcion en V
 78:     sei
 79: ret
 80:
 81: setup_ports:
 82:     clr conf
 83:     out input, conf ; PIND como entrada
 84:     ldi conf, 0x01
 85:     out ddrb, conf  ; PB0 como salida
 86: ret
 87:
 88: init_zp:
 89:     ldi zl, low (vector << 1)
 90:     ldi zh, high(vector << 1)
 91: ret
 92:
 93: timer_isr:
 94:     sbic led_port, led  ; si no esta apagado, lo apago
 95:     rjmp turn_off
 96:
 97:     sbi led_port, led   ; si no, lo prendo y salgo
 98: reti ; 1
 99:
100: turn_off:
101:     cbi led_port, led
102: reti ; 0
103:
104: vector: .db 0x00, 0x04, 0x03, 0x05
105:
```

8. Costos

A continuación se presenta un listado de los costos de los componentes utilizados en el práctico.

- Arduino UNO - $850
- Resistores - $50
- LED rojo - $20
- Pulsadores - $50
- Protoboard - $240
- Cables unipolares - $150

Sumando un costo total de $1360.

9. Resultados

Dada la frecuencia del cristal externo provisto por la placa Arduino UNO, de 16MHz, las frecuencias de trabajo se presentan en la Tabla 1, las cuales se calculan de acuerdo a (1), en donde la cantidad de bits utlizada para el cálculo es 16, de acuerdo a la cantidad de bits del timer 1.

$$f = \frac{2 \cdot f_{clock}}{prescale \cdot 2^{\#bits}} \tag{1}$$

PD0	PD1	Prescaler	Frecuencia [Hz]
0	0	No	0
0	1	64	7,63
1	0	256	1,91
1	1	1024	0,477

Tabla 1: Frecuencias de trabajo.

10. Conclusiones

El uso del timer brindó la facilidad de realizar una rutina cada un cierto tiempo, sin la necesidad de llamarla desde el código, permitiendo que esta se ejecute en cualquier parte del código. De esta forma fue posible actualizar la frecuencia de oscilación, sin que el el timer tenga que resetearse a mitad de un ciclo de conteo.