Trabajo Práctico N° 4 – Timers y PWM

Objetivos

Configuración y uso de timers.
Generación de interrupciones por eventos de timer
Manejo de antirrebotes de teclas.
Verificación de PWM a través de la variación de brillo de un LED

1. Timers

Descripción

Se pide hacer un programa que haga parpadear el LED conectado al PB0, en 3 frecuencias distintas o que lo deje ENCENDIDO FIJO, según los valores que haya en las entradas PD0 PD1 según se indica en la siguiente tabla:

PD0	PD1	ESTADO DEL LED
0	0	ENCENDIDO FIJO
0	1	PARPADEA CON PRESCALER CLK/64
1	0	PARPADEA CON PRESCALER CLK/256
1	1	PARPADEA CON PRESCALER CLK/1024

Nota: si alguien presiona las teclas mientras está funcionando el micro, los leds deberán cambiar acorde a la tabla.

Para resolver esta práctica usarán el **Timer1, interrupción por OVERFLOW**
Cuando el LED esté **ENCENDIDO FIJO** el timer estará apagado.
En los otros 3 casos, el timer contará los pulsos de clock divididos por prescaler 64, 256 y 1024 respectivamente.

Cuando se produce un overflow (desborde) deberán cambiar el estado del LED, es decir, si está prendido lo apagan y viceversa

Calcular la frecuencia o periodo con que se prenderá el LED en los 3 casos.

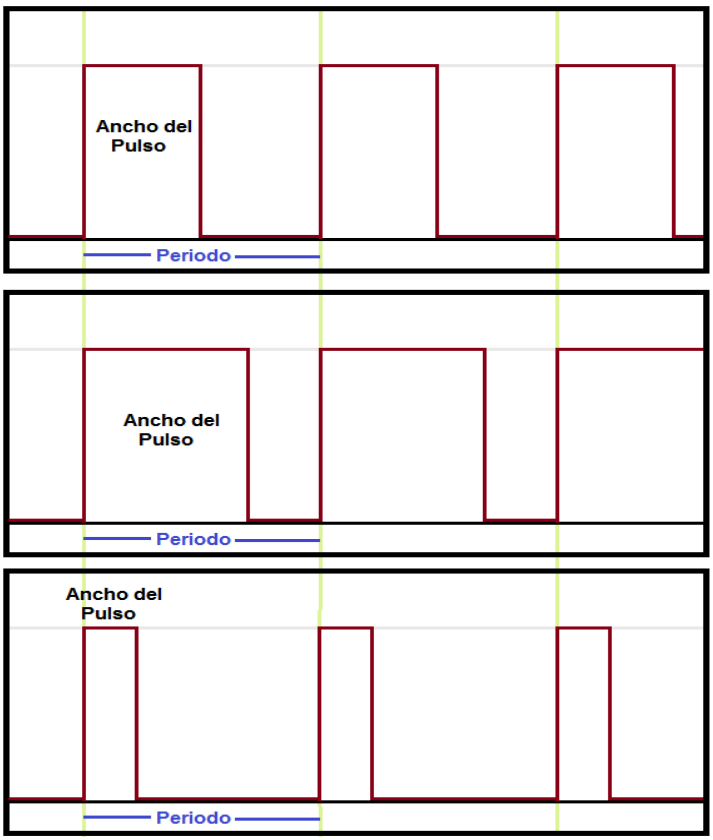
Se les recuerda que si usan la placa de Arduino, la frecuencia del clock es 16MHz.

En las entradas PD0 PD1 están conectados 2 switches, que como cualquier tecla produce rebotes al ser presionada. Deben implementar rutinas antirrebote para detectar correctamente las teclas

2. PWM

Descripción

Hacer un programa que aumente y disminuya el brillo de un LED. Para eso se dispone de 2 pulsadores (UP, DOWN) y un LED como indica el esquema. Para la variación de brillo se deberá usar PWM, o modulación por ancho de pulso, que consiste en modificar el ciclo de trabajo de una señal (sin modificar su frecuencia). Con esta señal se alimentará el LED, de forma que el valor medio de la señal será proporcional al brillo del LED, a mayor ancho de pulso, más brillo.

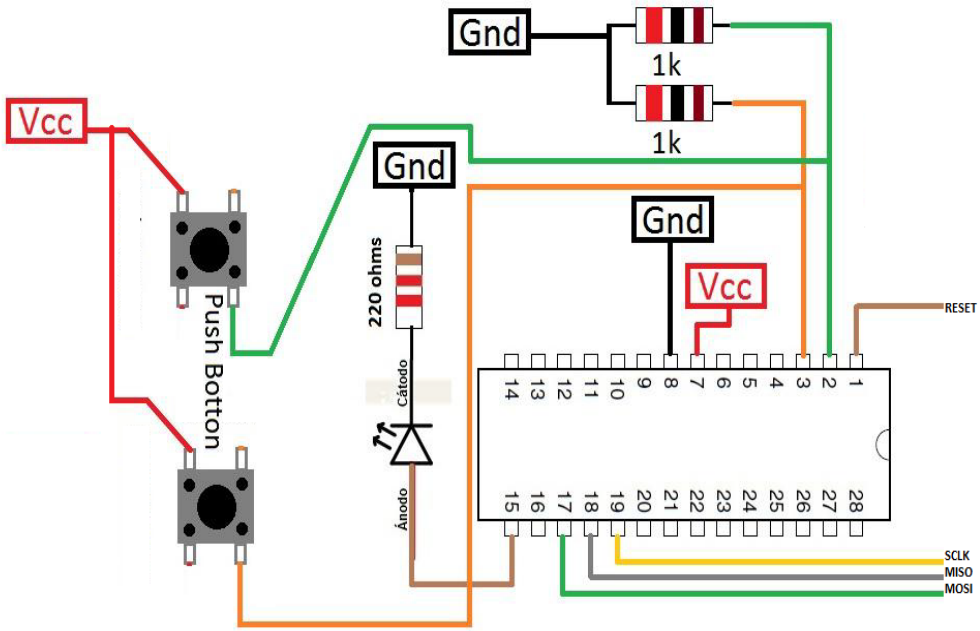


PWM

Modulación por ancho de pulso o PWM (Pulse Width Modulation), de una señal, es cuando se modifica el ciclo de trabajo o el ancho del pulso de una señal periódica, en este caso representado por una señal cuadrada, uno de los usos del PWM entre muchos otros, es controlar la cantidad de energía, en este caso el voltaje promedio es mayor conforme aumenta el ciclo de trabajo.

En la imagen se puede observar, que el período de la señal permanece fijo, por lo tanto, la frecuencia también, solamente cambia el ciclo de trabajo, en la primera se observa que el ciclo de trabajo es de aproximadamente 50% lo cual nos indica que es el porcentaje de voltaje promedio entregado a la carga.
El PWM se puede utilizar en varias cosas, como el control de la velocidad de motores de DC, la posición de un servomotor, fuentes conmutadas, entre otras cosas más.

Diagrama Esquemático



Materiales

1 LED
1 Resistencia de 220 Ohms
2 Pulsadores
2 Resistencias de 10Kohms
1 Microcontrolador ATmega8
Programador USBasp V3.0

Lectura recomendada

“ The AVR microcontroller and embedded systems. Embedded system using Assembly and C”. Autores: MUHAMMAD ALI MAZIDI, SARMAD NAIMI, SEPEHR NAIMI

Capítulos 4 y 16

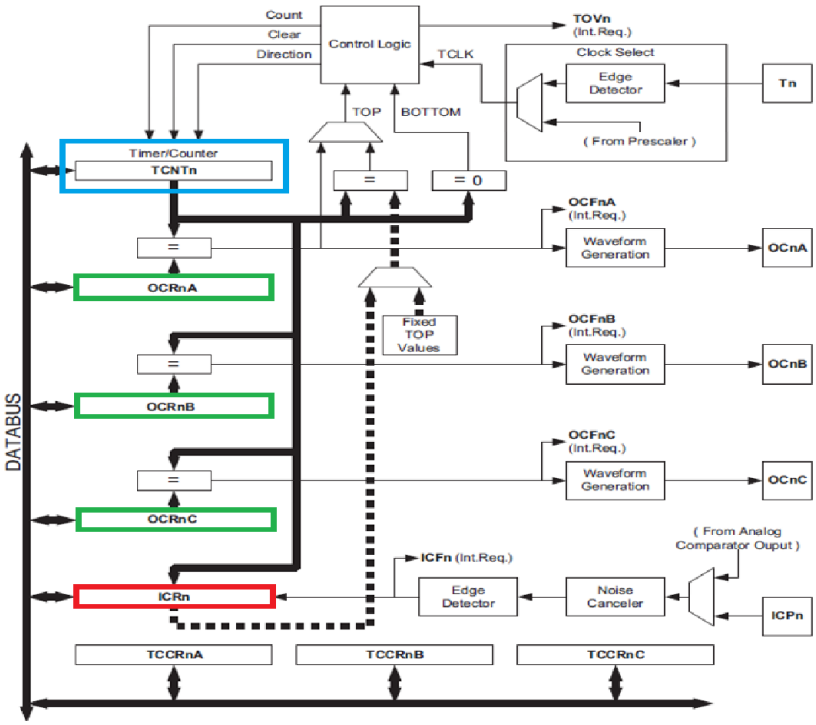
Temporización en AVR

El Atmega328p cuenta con dos timers de 8 bits (TC0 y TC2), con prescaler y modo de Comparación, y un timer de 16 bits (TC1) con prescaler y modos de Comparación y Captura. Estos permiten controlar de manera precisa los tiempos de ejecución de programa (gestión de eventos), así como también generar y medir ondas o señales. Cuenta además con seis canales de PWM, cuatro de 8 bits (asociados a TC0 y TC2) y dos de 16 bits (asociados a TC1).

El Atmega2560 cuenta con dos timers de 8 bits (TCO y TC2), con prescaler y modo de Comparación, y cuatro timers de 16 bits (TC1, TC3, TC4 y TC5) con prescaler y modos de Comparación y Captura. Estos permiten controlar de manera precisa los tiempos de ejecución de programa (gestión de eventos), así como también generar y medir ondas o señales. Cuenta además con cuatro canales de PWM de 8 bits (asociados a TC0 y TC2), y 12 canales de PWM con resolución programable de entre 2 y 16 bits (asociados a TC1, TC3, TC4 y TC5).

7.2.1. Timers de 16 bits

Para los timers TCn (con n=1 en Atmega328p y n=1, 3, 4 y 5 en Atmega2560), es decir, los timers de 16 bits, los registros principales del TCn son TCNTn, OCRnX (con X=A, B en Atmega328p y X=A, B, C en Atmega2560) e ICRn, todos de 16 bits.



El registro TCNTn contiene el valor del timer en cada instante (se puede escribir y leer).

Los registros OCRnX (Output Compare) se escriben con un valor que será comparado permanentemente con el valor de TCNTn. Los comparadores dan a su salida un '1' sólo en caso de coincidencia o "match" entre sus entradas, y dan un '0' para el resto del tiempo. La coincidencia levanta además un flag (OCnX) que puede usarse para interrupción.

El registro ICRn (Input Capture) "congela" el valor de TCNTn en el momento en el cual se produce el evento configurado en el "Edge detector" (con el bit ICESn del registro TCCRnB) en el pin ICPn. A su vez, en ese momento se levanta un flag de captura ICFn, permitiendo ir a su rutina de servicio (si la correspondiente interrupción está habilitada) a leer el valor "congelado" en ICRn.

Existen distintos modos de operación para el TCn, los cuales se eligen escribiendo los bits WGM10:3 de los registros TCCRnX.

Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Table 20-7. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

Uso de los bits **COMnX**[0:1]:

Table 20-3. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

Table 20-4. Compare Output Mode, Fast PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

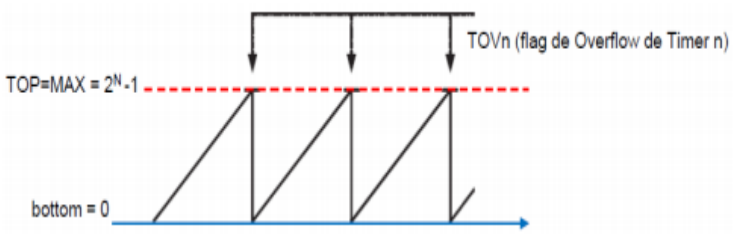
Table 20-5. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 9 or 11: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when down-counting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when down-counting.

Algunos de los modos de temporización para los timers de 16 bits son los siguientes:

MODO NORMAL (0)

El timer cuenta desde $\mathbf{0}$ hasta su valor máximo ($\mathbf{2^{16} - 1}$), desborda y vuelve a $\mathbf{0}$, levantando el flag de desborde TOVn. Es decir, desborda cada $\mathbf{2^{16}}$escalones del timer, cada uno de los cuales se mantiene durante un tiempo $Tcy = 1/FCPU$ e inmediatamente se incrementa.
Si se habilita la interrupción por desborde, podemos cambiar el estado de un bit en la rutina de servicio para verificar el tiempo transcurrido. Por ejemplo, para una frecuencia de reloj de 16MHz, con prescaler en 1, el tiempo entre desbordes será de:



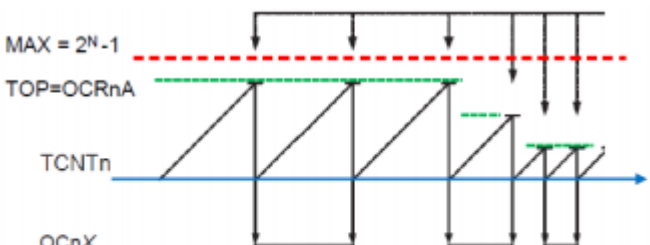
$$t = \frac{(2^{16})\ ciclos}{(16 * 10^6)\ ciclos/seg} = 4.096ms$$

Con prescaler en 8, el tiempo será de $8 * 4.096ms = 0.0328s$, etc

MODO CLEAR TIMER ON COMPARE MATCH o CTC (4)

El timer cuenta desde 0 hasta el valor TOP del registro OCRnX, se resetea y levanta el flag de coincidencia OCnX. Puede actuar también sobre el pin OCnX. Para lograr un período de N pulsos (N de 2 a 65536) debe hacerse

OCRnX = N-1.



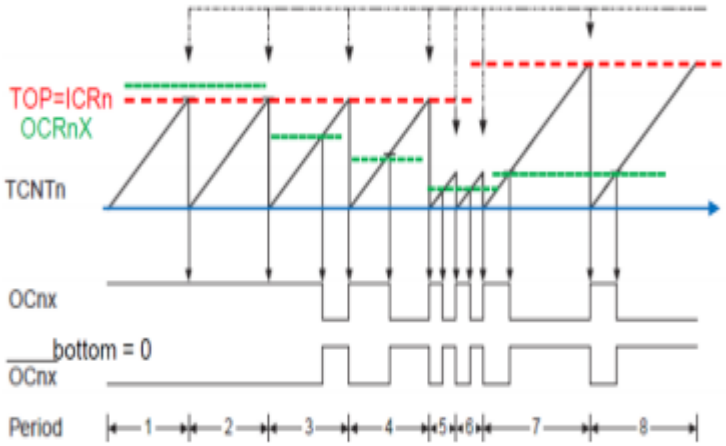
Si se habilita la interrupción por coincidencia, al igual que en el caso anterior, podemos cambiar el estado de un bit en la rutina de servicio para verificar el tiempo transcurrido. Pero una alternativa más fácil es activar el cambio de estado de OCnX en la configuración de los registros. Por ejemplo, para una frecuencia de reloj de 16MHz, con prescaler en 1, si se quiere lograr un tiempo entre coincidencias de 2.5ms, entonces:

$$\frac{(OCRnX + 1)\ ciclos}{(16 * 10^6)\ ciclos/seg} = 2.5ms$$

$$OCRnX = 39999$$

Con esta configuración y prescaler en 256, el tiempo será de $256 * 2.5ms = 0.64s$, etc.

MODO FAST PWM



El timer cuenta desde 0 hasta el valor TOP del registro ICRn y se resetea. Además, si la salida OCnX está configurada como “clear”, ésta se pone en ‘1’ al comienzo de cada período y se pone en ‘0’ en cada coincidencia del timer con ICRn. Si la misma se configura como “set”, actúa en forma opuesta. En definitiva, permite fijar el período con ICRn y el duty-cicle (tiempo en alto) con OCRnX.

Por ejemplo, para una frecuencia de reloj de 16MHz, con prescaler en 1, si se quiere lograr una señal de PWM al derecho, con un período fijo de 2.5ms, entonces:

$$\frac{(ICR1 + 1)\ ciclos}{(16 * 10^6)\ ciclos/seg} = 2.5ms$$

$$ICRn = 39999$$

Además, la salida de OCnX deberá estar configurada como “clear”.

MODO CORRECT PHASE PWM

El timer cuenta desde 0 hasta el valor TOP del registro ICRn y vuelve en forma descendente hasta 0 nuevamente. Además, la salida OCnX conmuta en cada coincidencia del timer con ICRn. Permite fijar el período con ICRn y el duty-cicle (tiempo en alto) con OCRnX.