

# **Principios de Mecatrónica – SDI-11561**

## **Ingeniería en Mecatrónica**

Hugo Rodríguez Cortés

Departamento de Ingeniería Eléctrica y Electrónica

Instituto Tecnológico Autónomo de México

Agosto 2023

- Una interrupción es una suspensión inmediata de la ejecución de un programa, enviando al contador del programa a otra acción/subrutina. Esta suspensión se produce, por ejemplo, cuando un periférico requiere la atención del procesador para realizar una operación de I/O.
- Las interrupciones son quizá el mecanismo más importante para la conexión del microcontrolador con el mundo exterior, sincronizando la ejecución de programas con acontecimientos externos/internos.
- Las interrupciones se pueden definir como: eventos que hacen que el microcontrolador deje de ejecutar la tarea que esta realizando para atender dicho acontecimiento y luego regresar a continuar la tarea que estaba realizando.

- Entonces, es un mecanismo que permite ejecutar un bloque de instrucciones, deteniendo la ejecución del programa y luego restablecer su ejecución sin afectarlo.
- En los microcontroladores AVR, para cada interrupción existe una Rutina de Servicio de Interrupción (ISR).
- Cuando se invoca una interrupción, el microcontrolador ejecuta el servicio de rutina de interrupción. En muchos microcontroladores, para cada interrupción hay una localidad fija en memoria que guarda la dirección de su ISR.
- En el AVR, el grupo de ubicaciones de memoria reservado para las direcciones de las ISR's se denomina tabla de vector de interrupciones.

Tras la activación de una interrupción, el microcontrolador realiza los siguientes pasos:

1. Finaliza la instrucción en ejecución y guarda la dirección de la siguiente instrucción (contador de programa) en el stack.
2. Brinca a una localidad fija en la memoria llamada Interrupted Vector Table. La tabla de vectores de interrupción dirige el microcontrolador a la dirección del ISR.
3. El microcontrolador empieza a ejecutar la “subrutina de servicio de interrupción” hasta que alcanza la última instrucción de la subrutina, la cual debe ser RETI (volver de la interrupción).
4. Al ejecutar la instrucción RETI, el microcontrolador regresa al lugar donde se interrumpió. Primero, obtiene la dirección del contador del programa (PC) y reinicia la ejecución desde esa dirección.

Fuente de interrupciones. Interrupciones para los Timers, Interrupciones externas en pines, Interrupciones de hardware interno, Interrupciones de periféricos (USART, SPI, ADC).

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x0002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A

13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

- Al resetear todas las interrupciones están deshabilitadas (enmascaradas) por lo que el microcontrolador no responderá a ellas. El bit D7 es el responsable de desenmascarar (habilitar) ó deshabilitar a todas las interrupciones.
- Pasos para habilitar una interrupción
  - ◆ Poner uno en D7 del SREG. La instrucción SEI lo realiza.

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

con I=1 en SREG, cada interrupción se habilita poniendo uno en su bandera de interrupción correspondiente.

## ■ Ejemplo, interrupciones de Timer0

**TIMSK0 – Timer/Counter Interrupt Mask Register**

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

cuando  $TOIE0 = 1$  e  $I = 1$  la interrupción de desbordamiento para el Timer/Counter0 se habilita.



## ■ Ejemplo, interrupciones en el periférico USART

**UCSRnA – USART Control and Status Register n A**

Bit	7	6	5	4	3	2	1	0	
	<b>RXCn</b>	<b>TXCn</b>	<b>UDREN</b>	<b>FEn</b>	<b>DORn</b>	<b>UPEn</b>	<b>U2Xn</b>	<b>MPCMn</b>	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

**UCSRnB – USART Control and Status Register n B**

Bit	7	6	5	4	3	2	1	0	
	<b>RXCIE n</b>	<b>TXCIE n</b>	<b>UDRIE n</b>	<b>RXEN n</b>	<b>TXEN n</b>	<b>UCSZ n2</b>	<b>RXB8 n</b>	<b>TXB8 n</b>	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

La bandera UDREN indica si el buffer de transmisión (UDRn) esta listo para recibir información. Si UDREN=1, el buffer esta vacío y por lo tanto listo para recibir información. La bandera UDREN puede generar una interrupción al escribir uno en UDRIEn. La interrupción queda habilitada con  $I = 1$ .

- Tras la activación de la interrupción, el propio AVR borra el bit I para asegurarse de que otra interrupción no pueda activarse. Al final de la ISR, la instrucción RETI fijará  $I = 1$  permitiendo que pueda entrar otra interrupción.

Como se sabe, la bandera de desbordamiento (TOV0) del timer0 se activa cuando el timer0 se desborda. Para detectar el desbordamiento se monitorea a la bandera del timer0 con la instrucción SBRS sobre TOV0. El problema con este método es que el microcontrolador queda atado al monitoreo de TOV0 sin poder hacer nada más.

Si la interrupción de desbordamiento TOIE0 del registro TIMSK0 se habilita, cuando el desbordamiento ocurra el microcontrolador saltará a la tabla de vectores de interrupción para dar servicio al ISR. De esta manera, el microcontrolador puede hacer otras cosas hasta que se le notifique que el temporizador se ha desbordado.

- Ejemplo, transferir datos de PORTC a PORTD y generar una onda cuadrada en PORTB.5.

```

.ORG      0x00                ;           —
          JMP      MAIN        ;           —
.ORG      0x0020              ;           —
          JMP      T0_OV_ISR   ;           —
.ORG      0x100                ;           —
MAIN:     LDI      R20, HIGH(RAMEND) ;       —
          OUT      SPH, R20     ;       —
          LDI      R20, LOW(RAMEND) ;       —
          OUT      SPL, R20     ;       —
          SBI      DDRB, 5      ;       —
          LDI      R20, (1<<TOIEO0) ;     —
          OUT      TIMSK0, R20  ;       —
          SEI                      ;       —
          LDI      R20, -32     ;       —
          OUT      TCNT0, R20   ;       —
          LDI      R20, 0x01    ;       —
          OUT      TCCR0, R20   ;       —
          LDI      R20, 0x00    ;       —
          OUT      DDRC, R20    ;       —
          LDI      R20, 0xFF    ;       —
          OUT      DDRD, R20    ;       —
HERE:     IN       R20, PINC     ;       —
          OUT      PORTD, R20   ;       —
          JMP      HERE         ;       —
.ORG      0x200                ;           —
T0_OV_ISR:: IN      R16, PORTB   ;       —
          LDI      R17, 0x20    ;       —
          EOR      R16, R17     ;       —
          OUT      PORTB, R16   ;       —
          LDI      R16, -32     ;       —
          OUT      TCNT0, R16   ;       —
          RETI                    ;       —

```

- Ejemplo, transferir datos de PORTC a PORTD y generar una onda cuadrada en PORTB.1 con Timer0 y PORTB.7 con Timer1.

```

.ORG    0x00                ;           —
        JMP      MAIN        ;           —
.ORG    0x001A              ;           —
        JMP      T1_OV_ISR   ;           —
.ORG    0x0020              ;           —
        JMP      T0_OV_ISR   ;           —
.ORG    0x100
MAIN:   LDI      R20, HIGH(RAMEND) ;           —
        OUT     SPH, R20        ;           —
        LDI     R20, LOW(RAMEND) ;           —
        OUT     SPL, R20        ;           —
        SBI     DDRB, 1         ;           —
        SBI     DDRB, 7         ;           —
        LDI     R20, (1<<TOIE00) ;           —
        OUT     TIMSK0, R20      ;           —
        LDI     R20, (1<<TOIE01) ;           —
        STS     TIMSK1, R20      ;           —
        SEI                     ;           —
        LDI     R20, -160        ;           —
        OUT     TCNT0, R20       ;           —
        LDI     R20, 0x01        ;           —
        OUT     TCCR0, R20       ;           —
        LDI     R20, HIGH(-640)  ;           —
        LDI     R21, LOW(-640)   ;           —
        STS     TCNT1H, R20      ;           —
        STS     TCNT1L, R21      ;           —
        LDI     R20, 0x00        ;           —
        LDI     R21, 0x01        ;           —
        STS     TCCR1A, R20      ;           —
        STS     TCCR1B, R21      ;           —

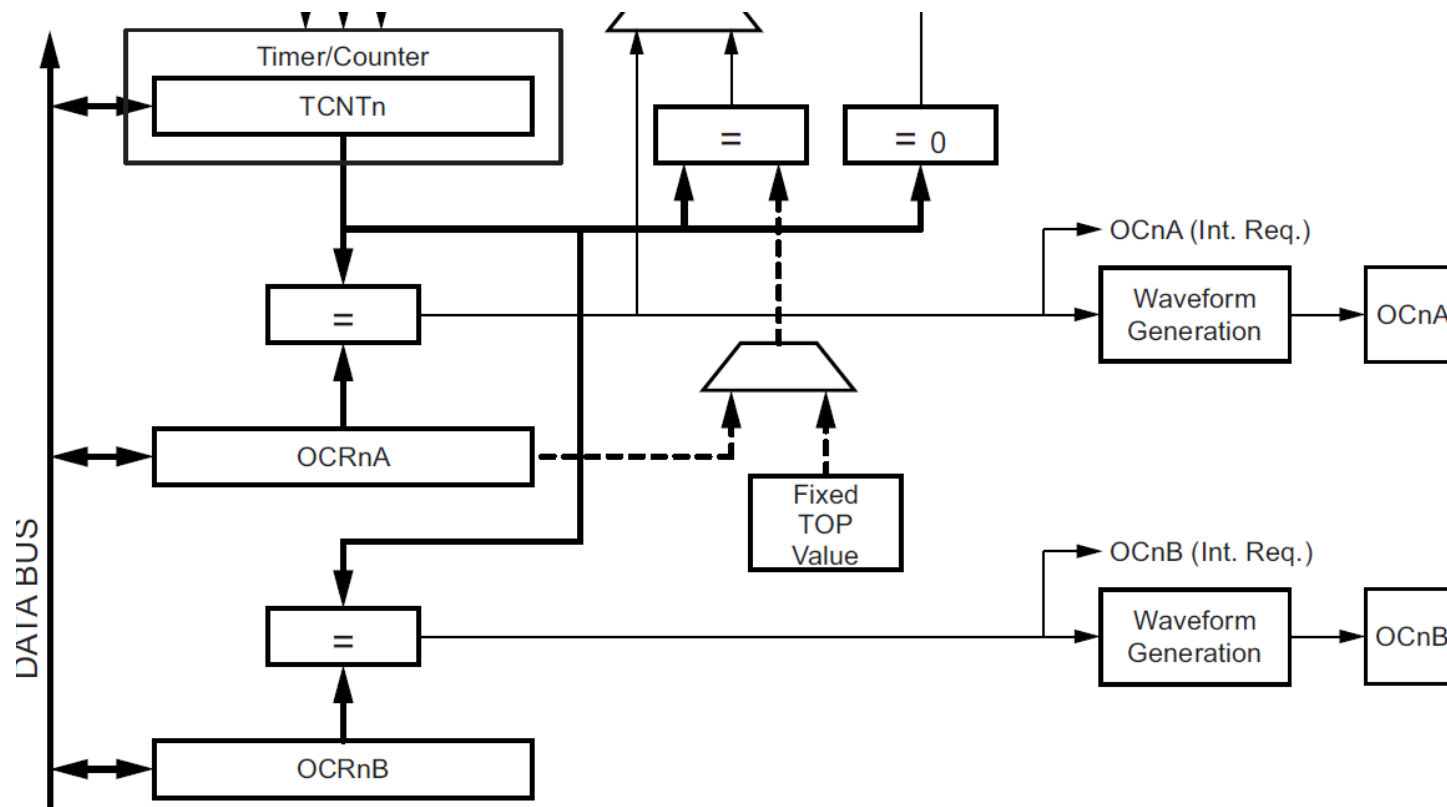
```

```

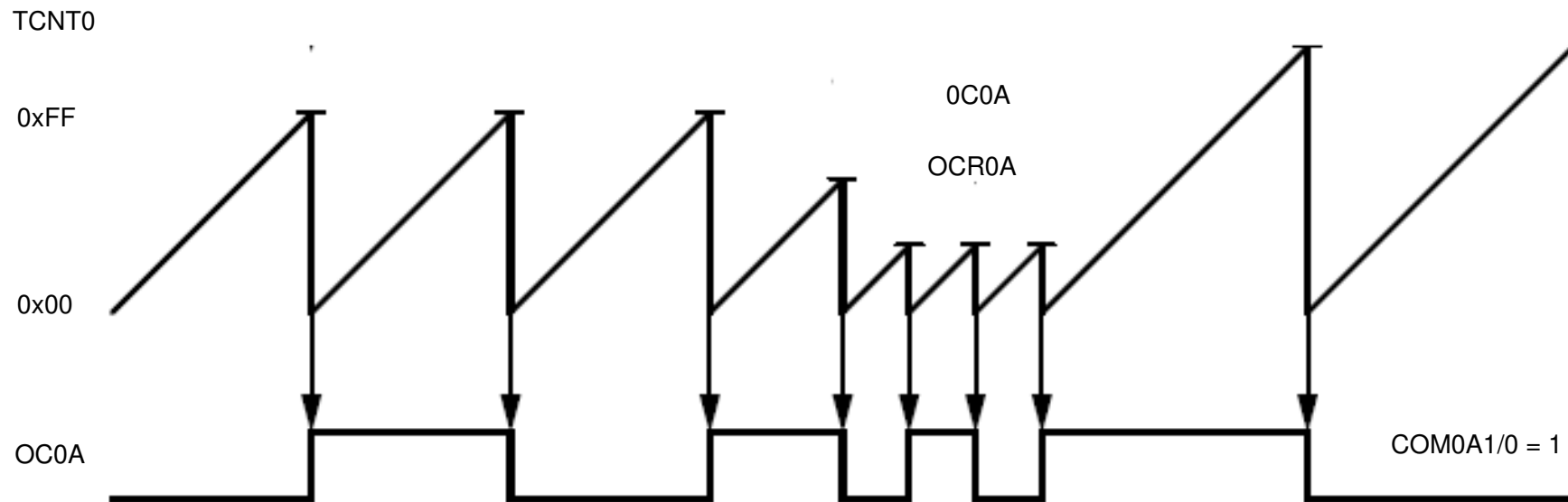
                                LDI    R20, 0x00      ;      —
                                LDI    R21, 0x01      ;      —
                                STS    TCCR1A, R20     ;      —
                                STS    TCCR1B, R21     ;      —
                                LDI    R20, 0x00      ;      —
                                OUT    DDRC, R20      ;      —
                                LDI    R21, 0xFF      ;      —
                                OUT    DDRD, R21      ;      —
HERE:                          IN     R20, PINC      ;      —
                                OUT    PORTD, R20     ;      —
                                JMP     HERE          ;      —
                                .ORG    0x200          ;      —
T0_OV_ISR::                   LDI    R16, -160       ;      —
                                OUT    TCNT0, R16     ;      —
                                IN     R16, PORTB      ;      —
                                LDI    R17, 0x02      ;      —
                                EOR    R16, R17       ;      —
                                OUT    PORTB, R16     ;      —
                                RETI                 ;      —
                                .ORG    0x300          ;      —
T1_OV_ISR::                   LDI    R18, HIGH(-640) ;      —
                                LDI    R19, LOW(-640) ;      —
                                STS    TCNT1H, R18     ;      —
                                STS    TCNT1L, R19     ;      —
                                IN     R18, PORTB      ;      —
                                LDI    R19, 0x80      ;      —
                                EOR    R18, R19       ;      —
                                OUT    PORTB, R18     ;      —
                                RETI                 ;      —

```

La familia de microcontroladores ATmega32 viene con al menos 3 timers, los cuales pueden ser usados como generadores de onda.



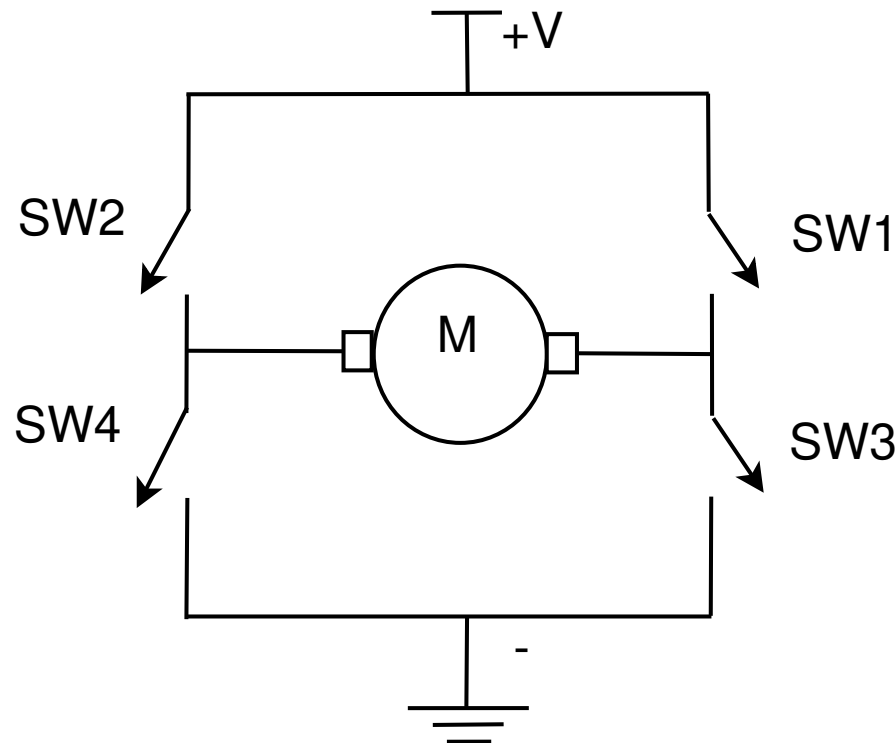
Generador de ondas cuadradas.



$$f_{OC0A} = \frac{f_{clk}}{2 \times K \times (1 + OCR0A)}$$

En un motor de corriente directa (DC), al invertir la polaridad de la alimentación se invierte la dirección de giro.

Con la ayuda de relevadores o algunos chips se puede implementar un puente H para invertir el giro de un motor de CD.

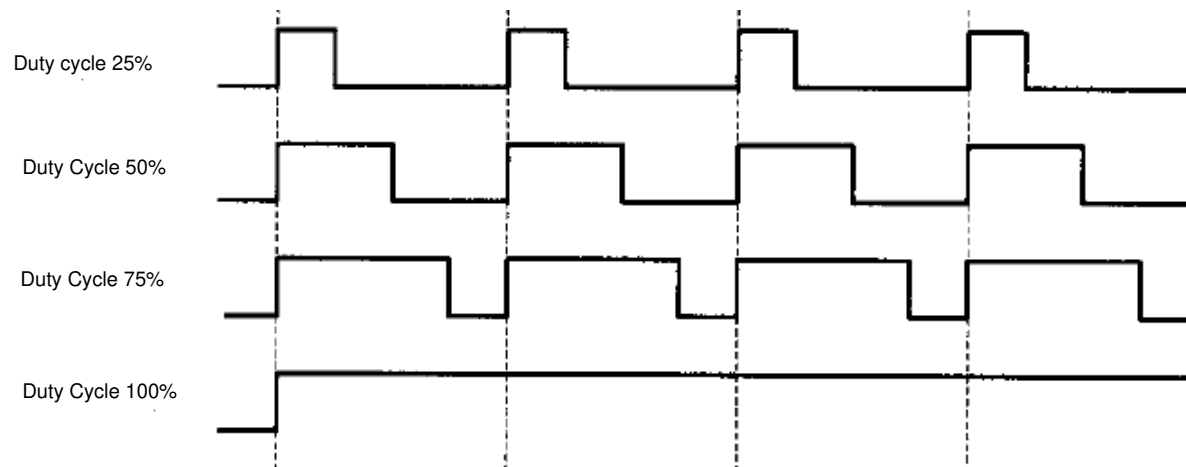




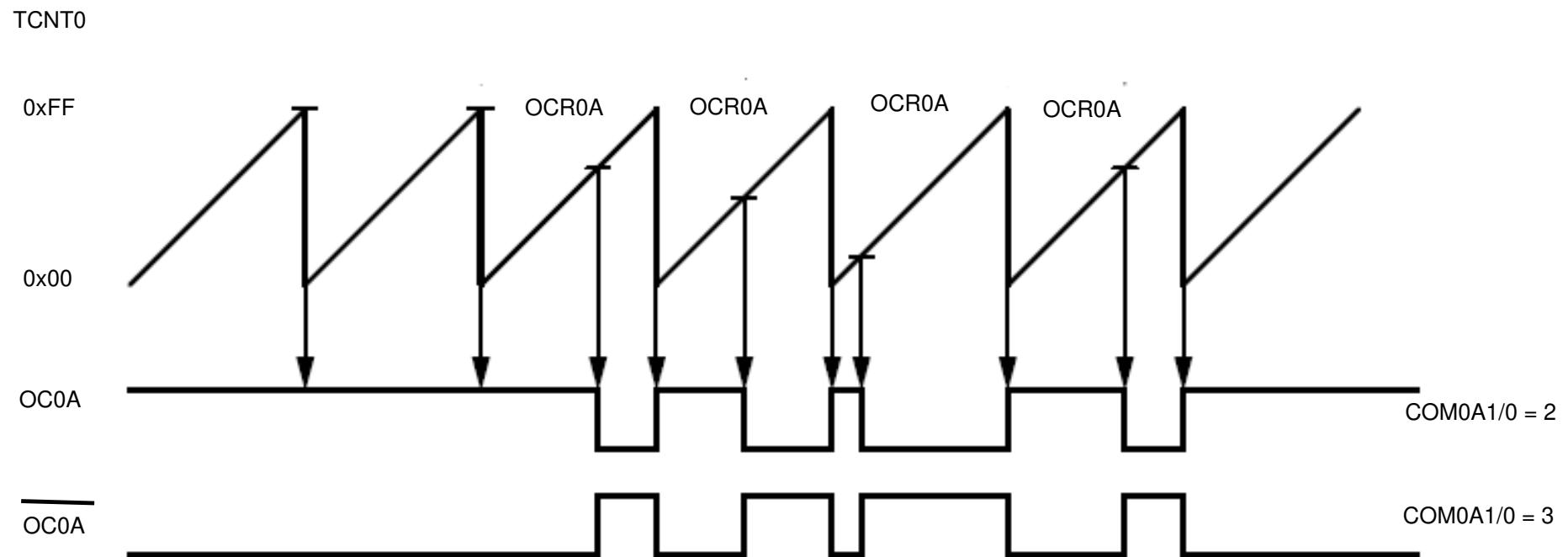
La velocidad del motor depende de tres factores: la carga conectada al motor, el voltaje y la corriente. Para una carga dada se puede mantener la velocidad utilizando un método llamada modulación por ancho de pulso (PWM).

Modulando el ancho de pulso se puede incrementar o disminuir la potencia alimentada al motor y por lo tanto modificar su velocidad.

Aunque el voltaje tiene amplitud constante tiene un ciclo de trabajo (duty cycle) diferente.

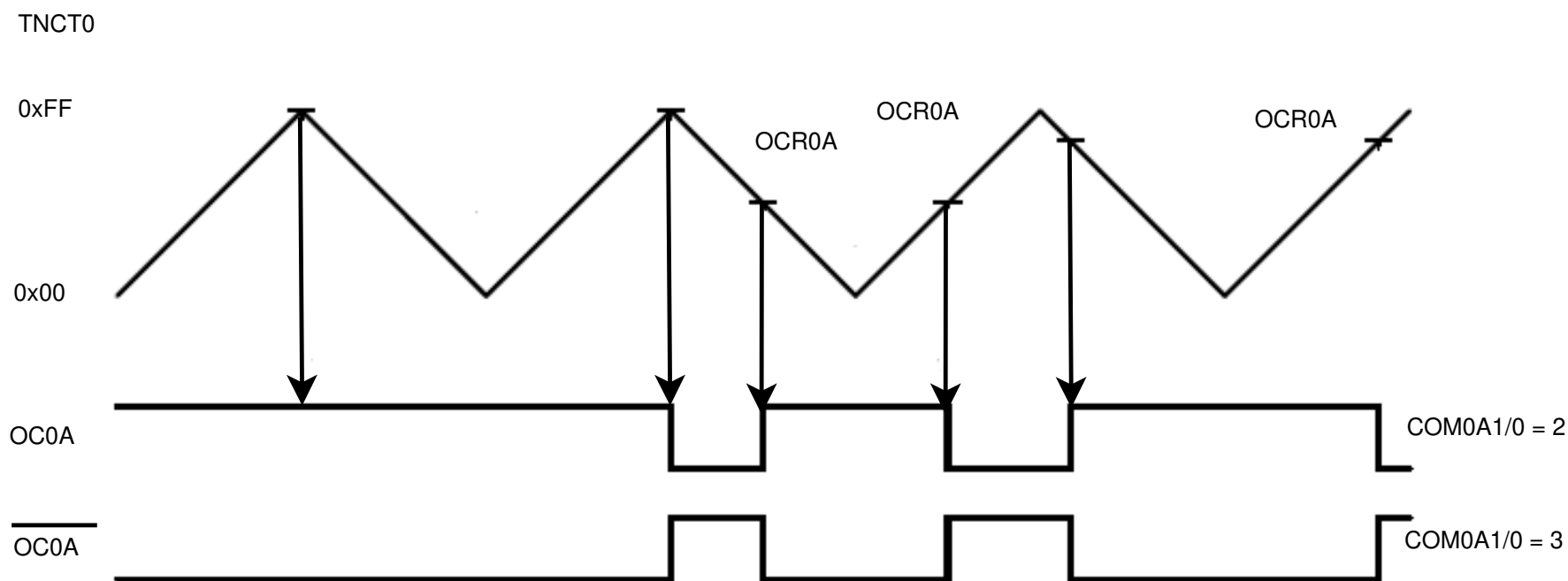


Se puede generar una señal de PWM de dos formas. PWM rápido (Fast PWM)



$$f_{OC0APWM} = \frac{f_{clk}}{K \times 256}, \quad DC^{NI} = \frac{OCR0A + 1}{256} \times 100, \quad DC^I = \frac{255 - OCR0A}{256} \times 100.$$

## PWM de fase correcta (Phase correct PWM)



$$f_{OC0APWM} = \frac{f_{clk}}{K \times 510}, \quad DC^{NI} = \frac{OCR0A}{255} \times 100, \quad DC^I = \frac{255 - OCR0A}{255} \times 100.$$

- La señal en el pin OC0A será visible si este se configura como salida.

**TCCR0A – Timer/Counter Control Register A**

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
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	

**TCCR0B – Timer/Counter Control Register B**

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at
0	0	0	0	Normal	0xFF	Immediate
1	0	0	1	PWM, phase correct	0xFF	TOP
2	0	1	0	CTC	OCRA	Immediate
3	0	1	1	Fast PWM	0xFF	BOTTOM
4	1	0	0	Reserved	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP
6	1	1	0	Reserved	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM

**Table 14-3. Compare Output Mode, Fast PWM Mode<sup>(1)</sup>**

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match, set OC0A at BOTTOM, (non-inverting mode).
1	1	Set OC0A on compare match, clear OC0A at BOTTOM, (inverting mode).

**Table 14-4. Compare Output Mode, Phase Correct PWM Mode<sup>(1)</sup>**

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down-counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down-counting.

Dibuje la forma de la señal generada.

```
SBI    DDRD, 6           ;      —
LDI    R22, 100          ;      —
OUT    OCR0A, R22        ;      —
LDI    R22, 0b01000000   ;      —
OUT    TCCR0A, R22       ;      —
LDI    R22, 0b00000001   ;      —
OUT    TCCR0B, R22       ;      —
HERE:  JMP    HERE       ;      —
```

Dibuje la forma de la señal generada.

	SBI	DDRD, 6	;	—
BEGIN:	LDI	R20, 69	;	—
	OUT	OCR0A, R20	;	—
	LDI	R22, 0b00000010	;	—
	OUT	TCCR0A, R22	;	—
	LDI	R22, 0b00000001	;	—
	OUT	TCCR0B, R22	;	—
L1:	IN	R20, TIFR0	;	—
	SBRS	R20, OCIE0A	;	—
	RJMP	L1	;	—
	LDI	R16, 1<<OCIE0A	;	—
	OUT	TIFR0, R16	;	—
	LDI	R20, 99	;	—
	OUT	OCR0A, R20	;	—
	LDI	R20, 0b00000010	;	—
	OUT	TCCR0A, R22	;	—
	LDI	R22, 0b00000001	;	—
	LDS	TCCR0B, R22	;	—
L2:	IN	R20, TIFR0	;	—
	SBRS	R20, OCIE0A	;	—
	RJMP	L2	;	—
	LDI	R16, 1<<OCIE0A	;	—
	OUT	TIFR0, R16	;	—
	RJMP	BEGIN	;	—

Suponga  $XTAL = 16\text{ MHz}$ , utilizando el modo no invertido, escriba un programa para generar una onda de frecuencia de  $7812.5\text{ Hz}$  y un ciclo de trabajo de  $37.5\%$



Suponga  $XTAL = 16 \text{ MHz}$ , utilizando el modo no invertido, escriba un programa para generar una onda de frecuencia de  $7812.5 \text{ Hz}$  y un ciclo de trabajo de  $37.5\%$

$$7812.5 = \frac{16M}{256 \times K} \Rightarrow K = \frac{16M}{7812.5 \times 256} = 8 \text{ pre-escalamiento}$$

$$37.5 = 100 \times \frac{OCR0A + 1}{100} \Rightarrow OCR0A = 95$$

```

SBI    DDRD, 6          ;
LDI    R20, 95          ;
OUT    OCR0A, R20       ;
LDI    R22, 0b10000011  ;
OUT    TCCR0A, R22      ;
LDI    R22, 0b00000010  ;
OUT    TCCR0B, R22      ;
HERE:  JMP    HERE      ;
    
```

Suponga  $XTAL = 16 \text{ MHz}$ , utilizando el modo invertido, escriba un programa para generar una onda de frecuencia de  $122.5 \text{ Hz}$  y un ciclo de trabajo de  $87.5\%$  en modo fase correcta.

Suponga  $XTAL = 16 \text{ MHz}$ , utilizando el modo invertido, escriba un programa para generar una onda de frecuencia de  $122.5 \text{ Hz}$  y un ciclo de trabajo de  $87.5\%$  en modo fase correcta.

$$122.5 = \frac{16M}{510 \times K} \Rightarrow K = \frac{16M}{122.5 \times 510} = 256.10 \approx 256 \text{ pre-escalamiento}$$

$$87.5 = 100 \times \frac{255 - OCR0A}{255} \Rightarrow OCR0A = 32$$

```

SBI    DDRD, 6           ;
LDI    R20, 32           ;
OUT    OCR0A, R20        ;
LDI    R22, 0b11000001   ;
OUT    TCCR0A, R22       ;
LDI    R22, 0b00000100   ;
OUT    TCCR0B, R22       ;
HERE:  JMP    HERE       ;

```

- La fase en la onda generada por el PWM de fase correcta no cambia.
- En el modo no invertido del PWM rápido el ciclo de trabajo no puede ser cero. De forma similar en el modo invertido del PWM rápido el ciclo de trabajo no puede ser del 100%.
- En el PWM de fase correcta el ciclo de trabajo puede ser 0% o 100%. Para controlar motores se prefiere el PWM de fase correcta.
- El PWM rápido se prefiere para frecuencias altas.