Principios de Mecatrónica – SDI-11561
Ingeniería en Mecatrónica

Hugo Rodríguez Cortés

Departamento de Ingeniería Eléctrica y Electrónica

Instituto Tecnológico Autónomo de México

Agosto 2023

Interrupciones

- 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 interrupciónes 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.

Interrupciones

- 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 interrpción, 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.

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.

Interrupciones

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	0x002	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

Interrupciones

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

Interrupciones

- 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.

Interrupciones

- 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.

Interrupciones

- Ejemplo, interrupciones en el periférico USART

UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	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	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	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.

Interrupciones

- 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 resgistro TIMSK0 se habilita, cuando el desboradmiento 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.

Interrupciones

- 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                ;  —
```

Interrupciones

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

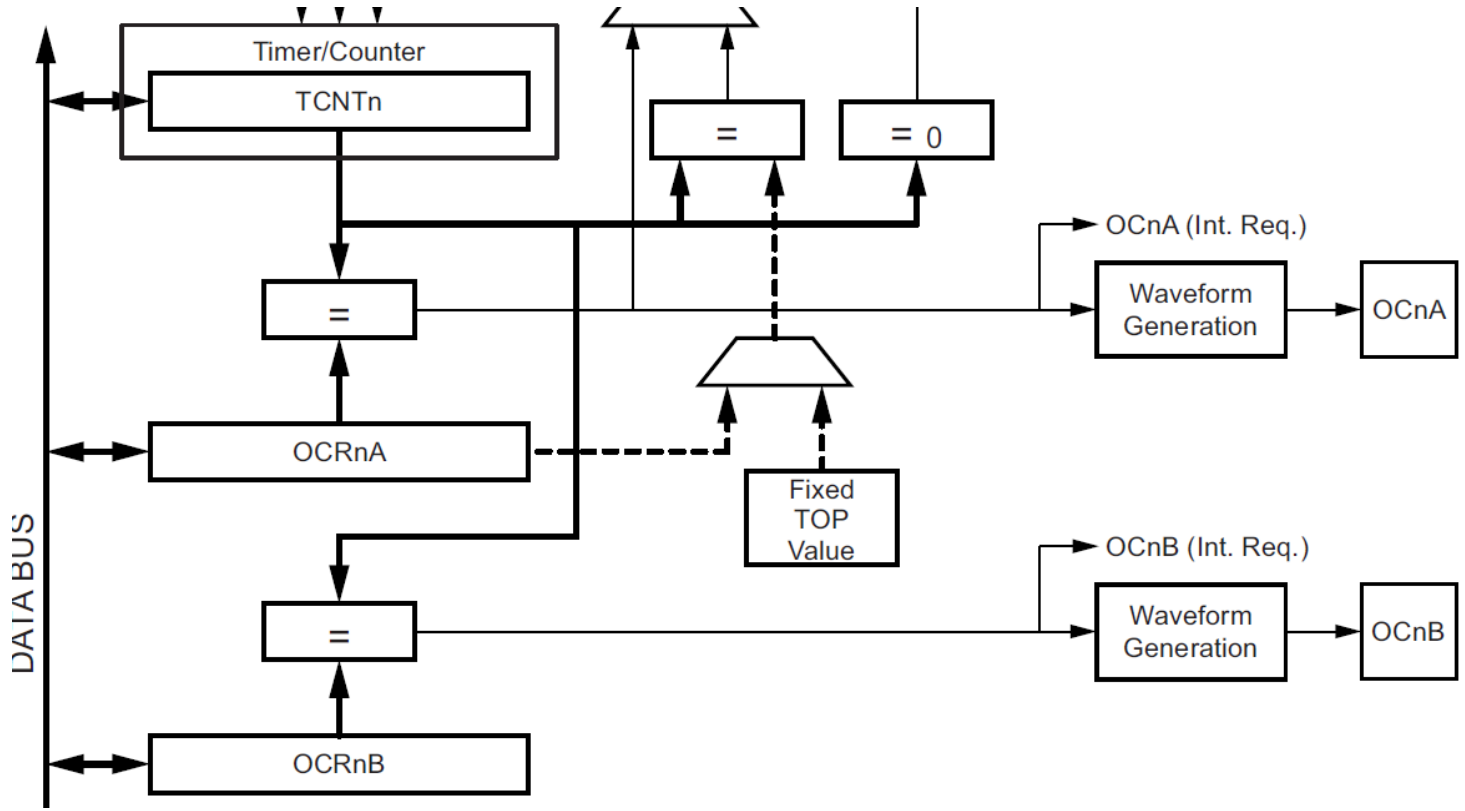
```
.ORG    0x00                    ;                   –
        JMP     MAIN            ;                   –
.ORG    0x001A                  ;                   –
        JMP     T1_OV_ISR       ;                   –
.ORG    0x0020                  ;                   –
        JMP     T0_OV_ISR       ;                   –
.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<<TOIEO0)    ;               –
        OUT     TIMSK0, R20     ;                   –
        LDI     R20, (1<<TOIEO1)    ;               –
        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     ;                   –
```

Interrupciones

```
                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                    ;   –
```

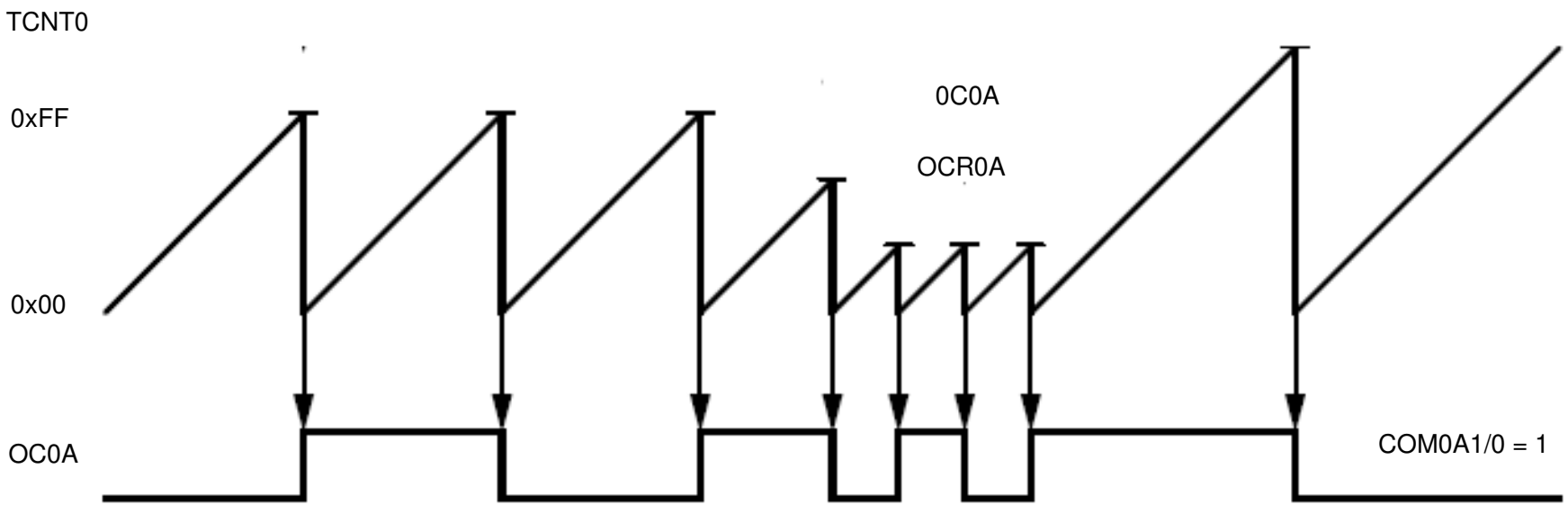
Generador de Ondas/PWM/Timer0

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



Generador de Ondas/PWM/Timer0

Generador de ondas cuadradas.

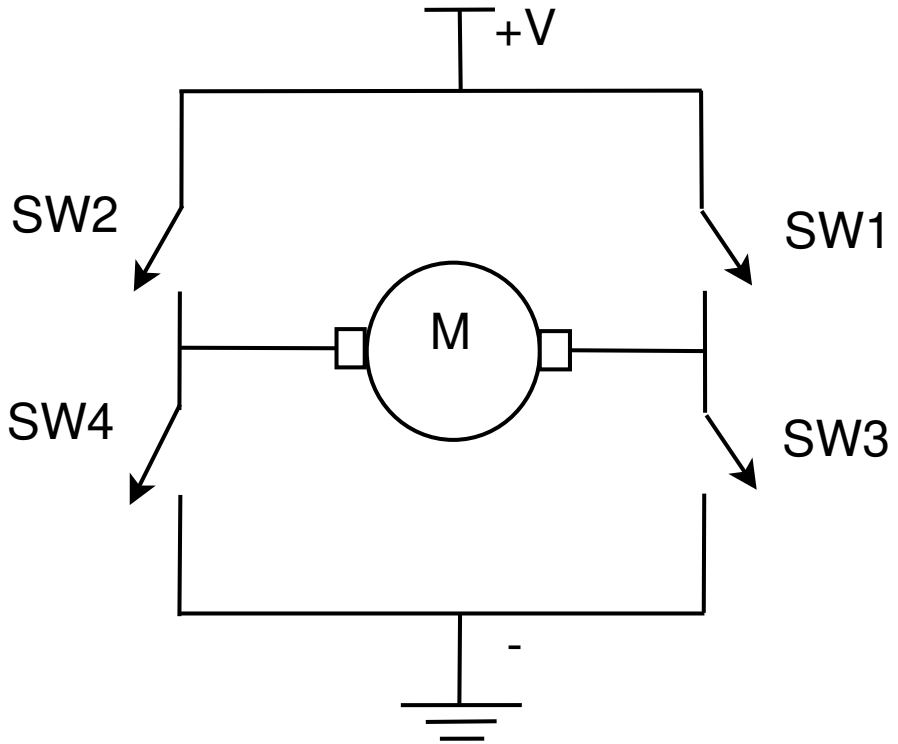


$$f_{\text{OC0A}} = \frac{f_{\text{clk}}}{2 \times K \times (1 + \text{OCR0A})}$$

PWM

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.

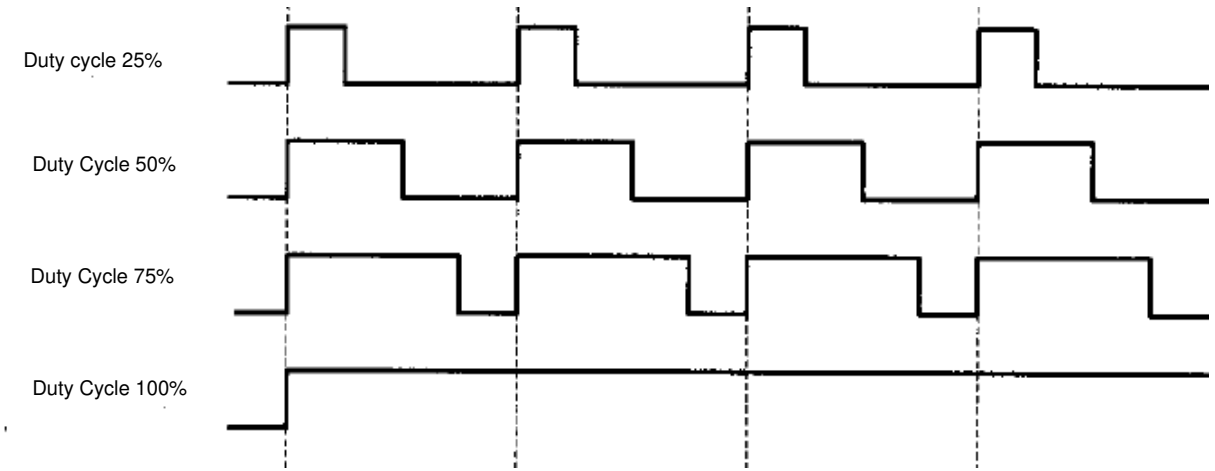


PWM

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).

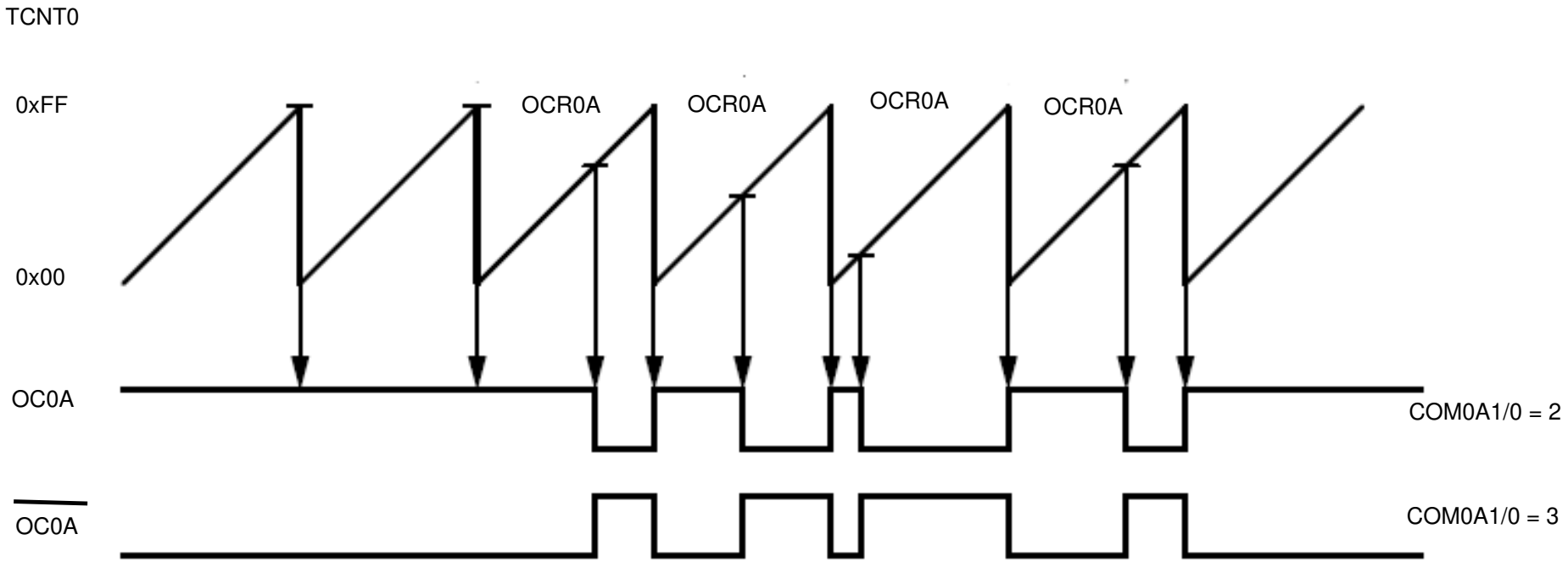
Modulando el ancho de pulso se puede incrementar o disminuir la potencia alimentada al motor y por lo tanto modificar su velocidad.

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



Generador de Ondas/PWM/Timer0

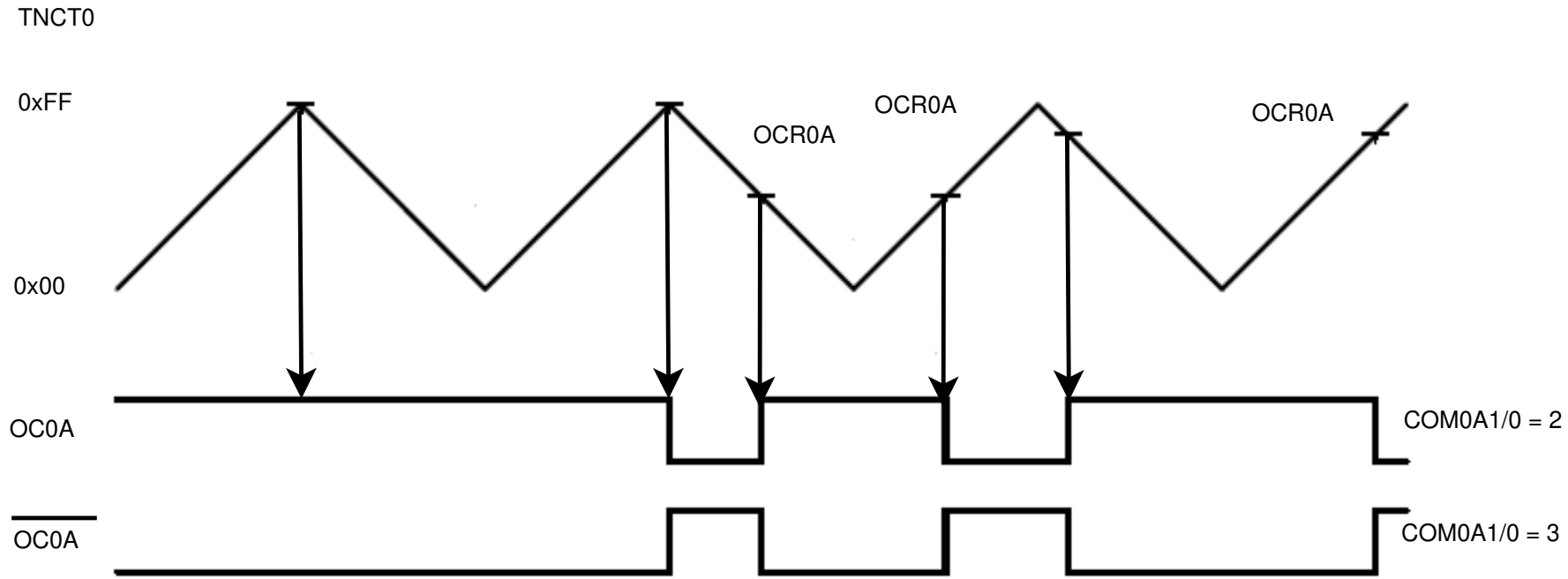
Se puede generar una señal de PWM de dos formas. PWM rápido (Fast PWM)



$$f_{\mathsf{OC0A_{PWM}}} = \frac{f_{\mathsf{clk}}}{K \times 256}, \; DC^{\mathsf{NI}} = \frac{\mathsf{OCR0A} + 1}{256} \times 100, \; DC^{\mathsf{I}} = \frac{255 - \mathsf{OCR0A}}{256} \times 100.$$

Generador de Ondas/PWM/Timer0

PWM de fase correcta (Phase correct PWM)



$$f_{\mathsf{OC0A_{PWM}}} = \frac{f_{\mathsf{clk}}}{K \times 510},\ DC^{\mathsf{NI}} = \frac{\mathsf{OCR0A}}{255} \times 100,\ DC^{\mathsf{I}} = \frac{255 - \mathsf{OCR0A}}{255} \times 100.$$

Generador de Ondas/PWM/Timer0

- 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	

- El modo de operación queda definido por la combinación de los bits del Waveform Generation Mode (WGM02, WGM01, WGM00) y los bits de Compare Output Mode (COM0A1, COM0A0).
- Los bits (COM0A1, COM0A0) controlan si la señal PWM es invertida o no, en el modo CTC, controlan si la salida debe ponerse en uno, en cero o alternarse cuando la comparación coincide.

Generador de Ondas/PWM/Timer0

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

Generador de Ondas/PWM/Timer0

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 14-3. Compare Output Mode, Fast PWM Mode(1)

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(1)

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.

Generador de Ondas/Timer0

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             ;  –
```

Generador de Ondas/Timer0

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            ;  –
```

PWM/Timer0

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

PWM/Timer0

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

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

$$37.5 = 100 \times \frac{\text{OCR0A} + 1}{256} \Rightarrow \text{OCR0A} = 95$$

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

PWM/Timer0

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

PWM/Timer0

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

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

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

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

PWM/Timer0

- 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 en regulación de potencia, rectificación y convertidores ddigital-analógico. Alta frecuencia permite utilizar componentes, inductores, capacitores de menores dimensiones.

ADC

- El microcontrolador utiliza valores discretos binarias, no obstante el mundo real es continuo (analógico).

- Los Convertidores Analógico a Digital (ADC) son la interface entre el mundo real, monitoreado por sensores, y el microcontrolador.

- Características de los ADC
  - **Resolución.** La cantidad más pequeña (step size) que puede detectar un ADC se especifica como n-bit, con n $= 8, 10, 12, 16, 24$. Se tiene un cierto control sobre el step size a través del voltaje de referencia V_{ref}.
  - **Tiempo de conversión.** El tiempo que tarda el ADC en transformar la entrada analógica a un número binario.

ADC

- Características de los ADC
  - **V_{ref}.** Es un voltaje de entrada que se utiliza como referencia. Este voltaje y la resolución en bits del ADC definen el step size. Dado V_{ref} para un ADC 8-bits el step size es $V_{\text{ref}}/256$.
  - **Dato digital.** En un ADC de 8-bit la salida digital es D7-D0, mientras que en un ADC de 10-bit la salida es D9-D0. Para calcular el voltaje detectado por el ADC se tiene

$$D_{\text{out}} = \frac{V_{\text{in}}}{\text{step size}}$$

con D_{out} en decimal, V_{in} el voltaje analógico de entrada.

ADC

- Ejemplo. En un ADC de 8-bit, se tiene $V_{\text{ref}} = 2.56\text{V}$. Calcular la salida digital si la entrada analógica es (a) 1.7V, (b) 2.1 V.

ADC

- Ejemplo. En un ADC de 8-bit, se tiene $V_{\text{ref}} = 2.56\text{V}$. Calcular la salida digital si la entrada analógica es (a) 1.7V, (b) 2.1 V.

- El step size es 2.56/256 = 10mV. Por lo tanto,
  - $D_{\text{out}} = 1.7/0.01 = 170 = 0b10101010.$
  - $D_{\text{out}} = 2.1/0.01 = 210 = 0b11010010.$
- ¿ Si el ADC es de 10 bit ?

ADC

- Ejemplo. En un ADC de 8-bit, se tiene $V_{\text{ref}} = 2.56\text{V}$. Calcular la salida digital si la entrada analógica es (a) 1.7V, (b) 2.1 V.

- El step size es 2.56/256 = 10mV. Por lo tanto,
  - $D_{\text{out}} = 1.7/0.01 = 170 = 0b10101010.$
  - $D_{\text{out}} = 2.1/0.01 = 210 = 0b11010010.$
- ¿ Si el ADC es de 10 bit ?

- El step size es 2.56/1024 = 2.5mV. Por lo tanto,
  - $D_{\text{out}} = 1.7/0.0025 = 680 = 0b1010101000.$
  - $D_{\text{out}} = 2.1/0.0025 = 840 = 0b1101001000.$

ADC

- En los AVR se tienen múltiples entradas analógicas y una sola salida digital por lo que se tienen señales de un inicio de conversión (SC) y final de conversión (EOC). Se utiliza el método de aproximación sucesiva para la conversión. Este método tiene tres componentes (a) registro de aproximación sucesiva (SAR), (b) comparador y (c) unidad de control.

- Asumiendo step size $= 10$ mV una aproximación sucesiva en un ADC de 8-bits para convertir una entrada analógia de 1.0V sigue los siguientes pasos

Binario	Decimal	Compara	Binario
1000 0000	1.28 V	$1.28 > 1.0$	0000 0000
0100 0000	0.64 V	$0.64 < 1.0$	0100 0000
0110 0000	0.96 V	$0.96 < 1.0$	0110 0000
0111 0000	1.12 V	$1.12 > 1.0$	0110 0000
0110 1000	1.08 V	$1.08 > 1.0$	0110 0000
0110 0100	1.0 V	$1.0 = 1.0$	0110 0100
0110 0110	1.02 V	$1.02 > 1.0$	0110 0100
0110 0101	1.01 V	$1.01 > 1.0$	0110 0100

ADC/ATMega328P

Features

- 10-bit resolution
- 0.5 LSB integral non-linearity
- ±2 LSB absolute accuracy
- 65 to 260µs conversion time
- Up to 15kSPS
- 6 multiplexed single ended input channels
- 2 additional multiplexed single ended input channels
- Temperature sensor input channel
- Optional left adjustment for ADC result readout
- 0 to V_{CC} ADC input voltage range
- Selectable 1.1V ADC reference voltage
- Free running or single conversion mode
- Interrupt on ADC conversion complete
- Sleep mode noise canceler

ADC/ATMega328P

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

(REF1, REF0) Seleccionan el voltaje de referencia. (ADLAR) Selecciona el formato de la salida digital. (MUX3, MUX2, MUX1, MUX0) Seleccionan las entradas analógicas conectadas al ADC.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

ADC/ATMega328P

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADC/ATMega328P

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8(1)
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

ADC/ATMega328P

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

(ADEN) Habilita/deshabilta al ADC. (ADSC) Inicio de conversión. (ADATE) Inicia la conversión en el flanco de subida de la señal seleccionada. (ADIF) Bandera para interrupción, es uno cuando se completa la conversión. (ADIE) Habilita la interrupción. (ADPS2, ADPS1, ADPS0) Escalamiento de la frecuencia del reloj de entrada al ADC.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC/ATMega328P

- El algoritmo de aproximación sucesiva requiere de un pulso de reloj entre 50kHz y 200kHz para obtener la máxima resolución. Si se requiere una resolución menor a 10-bits la frecuencia puede ser mayor.

- Una conversión normal requiere 13 ciclos de reloj, la primera conversión, al inicializar los circuitos, requiere de 25 ciclos de reloj.

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5

ADC/ATMega32P

Analice el siguiente código.

```
                LDI     R16, 0xFF       ;               –
                OUT     DDRB, R16       ;               –
                OUT     DDRC, R16       ;               –
                LDI     R16, 0          ;               –
                OUT     DDRA, R16       ;               –
                LDI     R16, 0x87       ;               –
                STS     ADCSRA, R16     ;               –
                LDI     R16, 0xC0       ;               –
                STS     ADMUX, R16      ;               –
READ_ADC:                               ;               –
                CBR     ADCSRA, ADSC    ;               –
KEEP_POLLING:                           ;               –
                SBRS    ADCSRA, ADSC    ;               –
                RJMP    KEEP_POLLING    ;               –
                CBR     ADCSRA, ADIF    ;               –
                LDS     R16, ADCL       ;               –
                OUT     PORTD, R16      ;               –
                LDS     R16, ADCH       ;               –
                OUT     PORTB, R16      ;               –
                RJMP    READ_ADC        ;               –
```

ADC/ATMega32P

Analice el siguiente código.

```
.CSEG                                                   ;   –
                RJMP        MAIN                        ;   –
.ORG            ADCCaddr                                ;   –
                RJMP        ADC_INT_HANDLER             ;   –
.ORG            40                                      ;   –

MAIN:           LDI         R16, HIGH(RAMEND)           ;   –
                OUT         SPH, R16                    ;   –
                LDI         R16, LOW(RAMEND)            ;   –
                OUT         SPL, R16                    ;   –
                SEI                                     ;   –
                LDI         R16, 0xFF                   ;   –
                OUT         DDRB, R16                   ;   –
                OUT         DDRC, R16                   ;   –
                LDI         R16, 0                      ;   –
                OUT         DDRA, R16                   ;   –
                LDI         R16, 0x8F                   ;   –
                STS         ADCSRA, R16                 ;   –
                LDI         R16, 0xC0                   ;   –
                STS         ADMUX, R16                  ;   –
                CBR         ADCSRA, ADSC                ;   –
WAIT_HERE:                                              ;   –
                RJMP        WAIT_HERE                   ;   –

ADC_INT_HANDLER:                                        ;   –
                LDS         R16, ADCL                   ;   –
                OUT         PORTD, R16                  ;   –
                LDS         R16, ADCH                   ;   –
                OUT         PORTB, R16                  ;   –
                CBR         ADCSRA, ADSC                ;   –
                RETI                                    ;   –
```