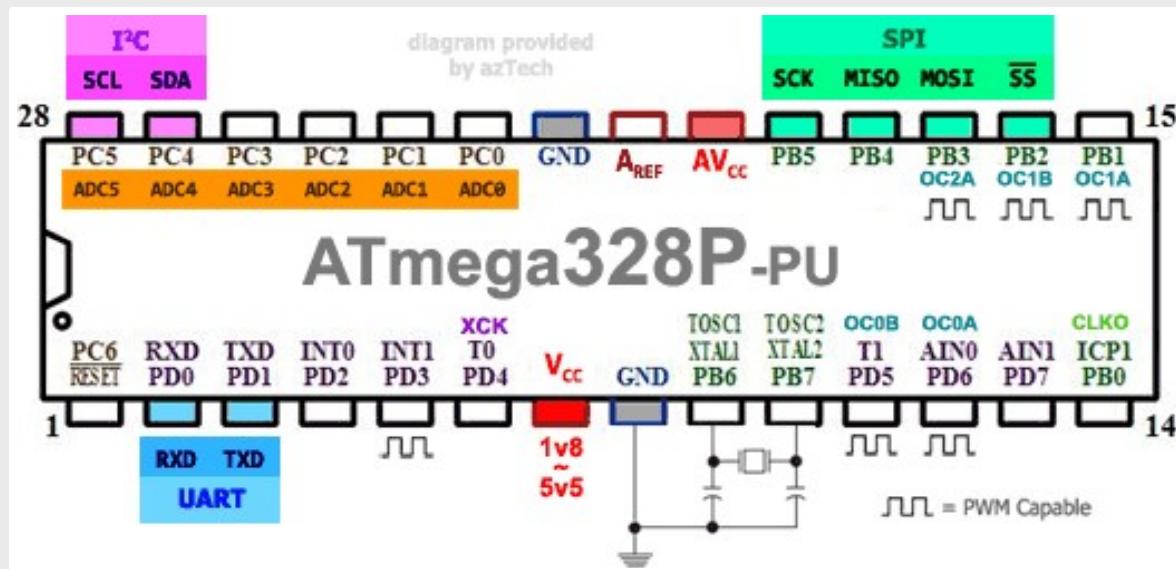
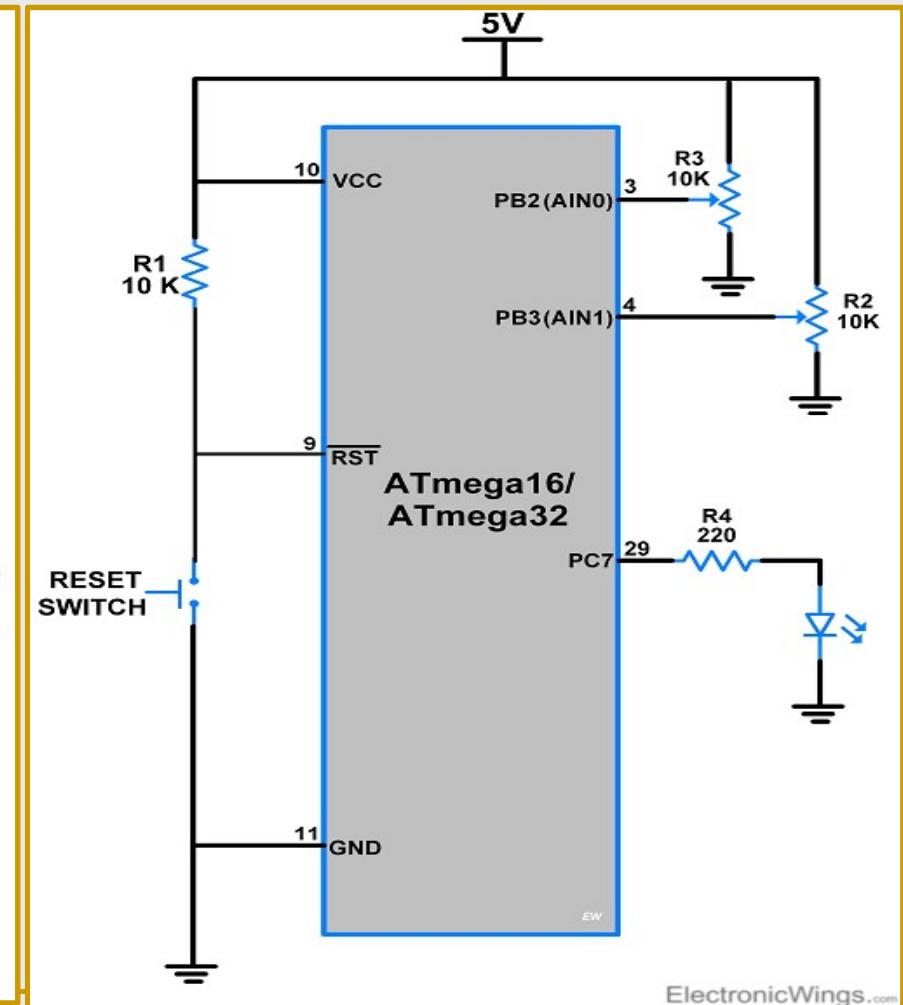


Teórica 6



Comparador analógico en AVR

(XCK/T0)	PB0	<input type="checkbox"/>	1	40	<input type="checkbox"/>	PA0	(ADC0)
(T1)	PB1	<input type="checkbox"/>	2	39	<input type="checkbox"/>	PA1	(ADC1)
Analog Comparator Pins	(INT2/AIN0)	PB2	<input type="checkbox"/>	3	<input type="checkbox"/>	PA2	(ADC2)
	(OC0/AIN1)	PB3	<input type="checkbox"/>	4	<input type="checkbox"/>	PA3	(ADC3)
	(SS)	PB4	<input type="checkbox"/>	5	<input type="checkbox"/>	PA4	(ADC4)
	(MOSI)	PB5	<input type="checkbox"/>	6	<input type="checkbox"/>	PA5	(ADC5)
	(MISO)	PB6	<input type="checkbox"/>	7	<input type="checkbox"/>	PA6	(ADC6)
	(SCK)	PB7	<input type="checkbox"/>	8	<input type="checkbox"/>	PA7	(ADC7)
	RESET		<input type="checkbox"/>	9	<input type="checkbox"/>	AREF	
	VCC		<input type="checkbox"/>	10	ATmega	31	<input type="checkbox"/> AGND
	GND		<input type="checkbox"/>	11	16 / 32	30	<input type="checkbox"/> AVCC
	XTAL2		<input type="checkbox"/>	12		29	<input type="checkbox"/> PC7 (TOCS2)
	XTAL1		<input type="checkbox"/>	13		28	<input type="checkbox"/> PC6 (TOCS1)
	(RXD)	PD0	<input type="checkbox"/>	14		27	<input type="checkbox"/> PC5 (TD1)
	(TXD)	PD1	<input type="checkbox"/>	15		26	<input type="checkbox"/> PC4 (TD0)
	(INT0)	PD2	<input type="checkbox"/>	16		25	<input type="checkbox"/> PC3 (TMS)
	(INT1)	PD3	<input type="checkbox"/>	17		24	<input type="checkbox"/> PC2 (TCK)
	(OC1B)	PD4	<input type="checkbox"/>	18		23	<input type="checkbox"/> PC1 (SDA)
	(OC1A)	PD5	<input type="checkbox"/>	19		22	<input type="checkbox"/> PC0 (SCL)
	(ICP1)	PD6	<input type="checkbox"/>	20		21	<input type="checkbox"/> PD7 (OC2)



Comparador analógico (AC)

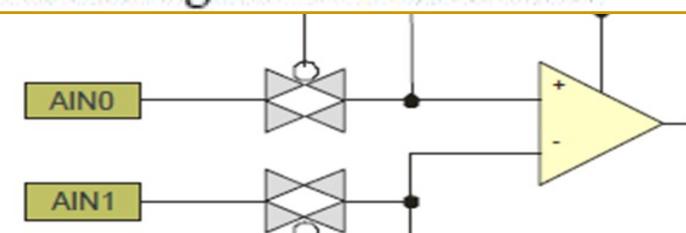
El comparador analógico es un recurso que indica la relación existente entre dos señales analógicas externas.

Útil para aplicaciones en donde no precisa conocer el valor digital de una señal analógica, sino que es suficiente con determinar si es mayor o menor que alguna referencia.

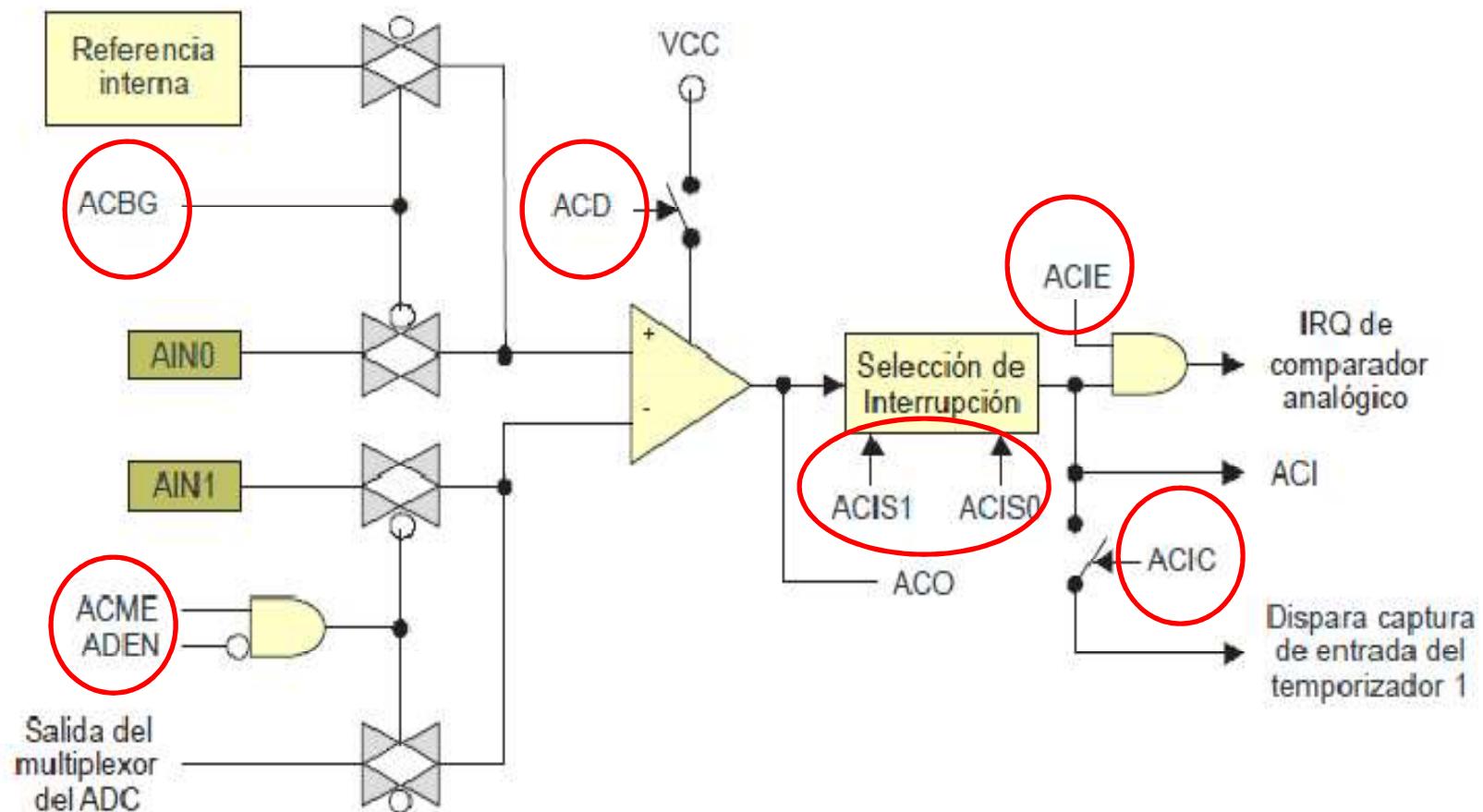
Se basa en un amplificador operacional en lazo abierto, el cual se va a saturar cuando una entrada analógica (**AIN0**) sea mayor que otra (**AIN1**), poniendo en alto al bit **ACO** (**ACO**, *Analog Comparator Output*).

Por medio de interruptores analógicos es posible remplazar la entrada **AIN0** por una referencia interna cuyo valor típico es de 1.1 V.

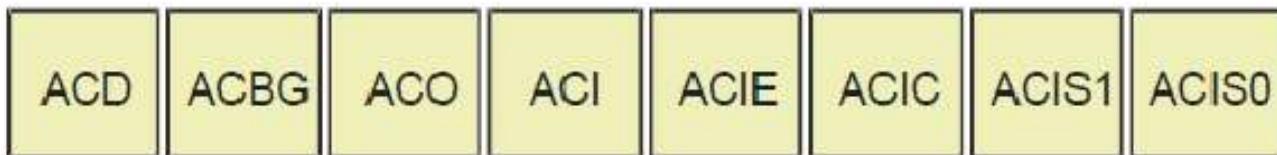
La entrada **AIN1** puede remplazarse por la salida del multiplexor del ADC, con ello es posible comparar más de una señal analógica con la misma referencia. El ADC debe estar deshabilitado.



Comparador analógico (AC)



Registro ACSR. Registro Control y Estado del AC.



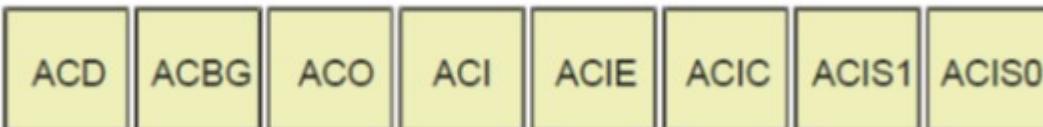
Bit 7 – ACD: AC Disable. Deshabilita el AC para minimizar el consumo de energía (con o el AC está activo). Si el AC está inactivo y se va a activar nuevamente, se sugiere deshabilitar su interrupción porque podría generar un evento.

Bit 6 – ACBG: AC Bandgap Select. Al ponerlo en alto, la entrada en la terminal positiva proviene de un voltaje de referencia interno (V_{BG}).

Bit 5 – ACO: AC Output. La salida del AC se sincroniza y se muestra en este bit, la sincronización toma 1 ó 2 ciclos de reloj.

Bit 4 – ACI: AC Interrupt Flag. Bandera para generar interrupción. Su activación está en función del resultado del comparador y de los bits **ACIS1** y **ACISO**.

ACSR.



Bit 3 – ACIE: AC Interrupt Enable. Habilitador de interrupciones del comparador analógico.

Bit 2 – ACIC: AC Input Capture Enable. Habilita la función de captura de entrada del temporizador 1. La salida del comparador se conecta al hardware de captura, utilizando sus recursos de selección de flanco y cancelación de ruido. Para que el comparador dispare la interrupción por captura de entrada, el bit TCIE del registro TIMSK debe estar en alto.

Bits 1 y 0 – ACIS[1:0]: AC Interrupt Mode Select. Seleccionan el modo para la interrupción. Al modificar estos bits, se sugiere deshabilitar la interrupción por que se podría generar eventos no deseados.

ACIS1	ACISO	Modo
0	0	Interrupción por commutación
0	1	Reservado
1	0	Flanco de bajada en ACo
1	1	Flanco de subida en ACo

Registro DIDR1. Registro 1 para deshabilitar la entrada digital.



Bits 7 al 2: No están implementados.

Bits 1 y 0 – AIN1D y AIN2D. Deshabilitan la entrada digital. Las terminales digitales incluyen un *buffer de entrada*, con un 1 en estos bits el buffer queda inhabilitado y por lo tanto, una entrada digital siempre se leería con un 0.

Se sugiere escribir un 1 en estos si se usa el comparador analógico para reducir el consumo de potencia.

El bit **ACME** se encuentra en el Registro ADCSRB (Registro B de Control y Estado del ADC).



Bits ACME (AC Multiplexer Enable). Al ser puesto en alto, la entrada negativa del comparador proviene de la salida del multiplexor del ADC.

El ADC debe estar deshabilitado y el canal de entrada analógica se selecciona con los bits **MUX[3:0]** del registro **ADMUX**.

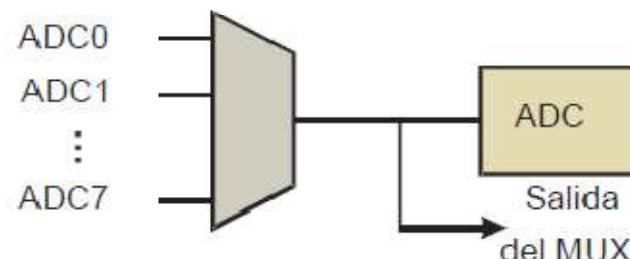
Si el ADC se habilita, aunque **ACME** tenga 1 la entrada será proporcionada en la terminal **AIN1**.

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
ADCSRB	-	ACME	-	-	-	ADTS2	ADTS1	ADTS0		(\$7B)

La entrada **AIN0** se puede reemplazar por un valor constante (V_{BG}). El bit **ACBG** permite hacer esta selección.

	Mínimo	Típico	Máximo
Referencia Interna (V_{BG})	1.0	1.1	1.2

La entrada **AIN1** se puede reemplazar por una de las entradas para conversión analógico-digital.



Para ello se requiere que:

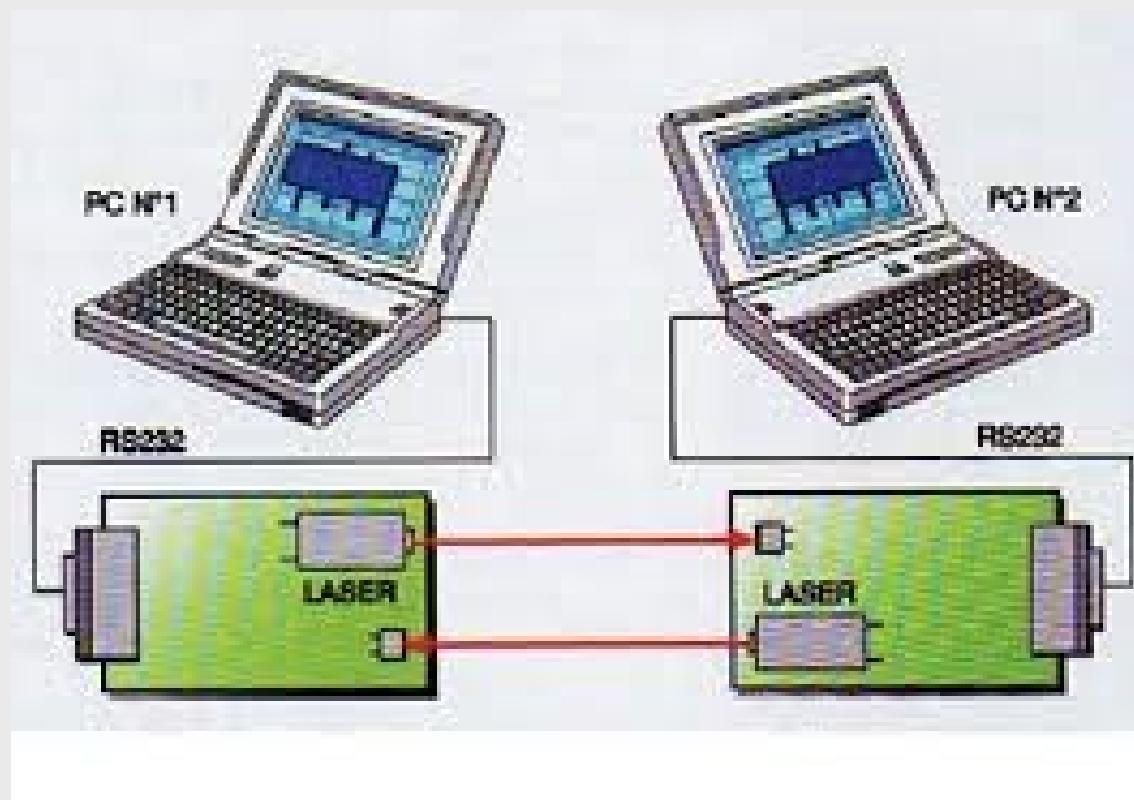
- El ADC esté deshabilitado (**ADEN** = 0).
- El MUX se habilite para ser usado por el comparador (**ACME** = 1).

Bit	7	6	5	4	3	2	1	0	
Read/Write	R/W	ADCSRA							
Initial Value	0	0	0	0	0	0	0	0	

INTERFACES PARA UNA COMUNICACIÓN SERIAL

- Comunicación serial a través de la USART
- Comunicación serial por SPI
- Comunicación serial via TWI –I2C

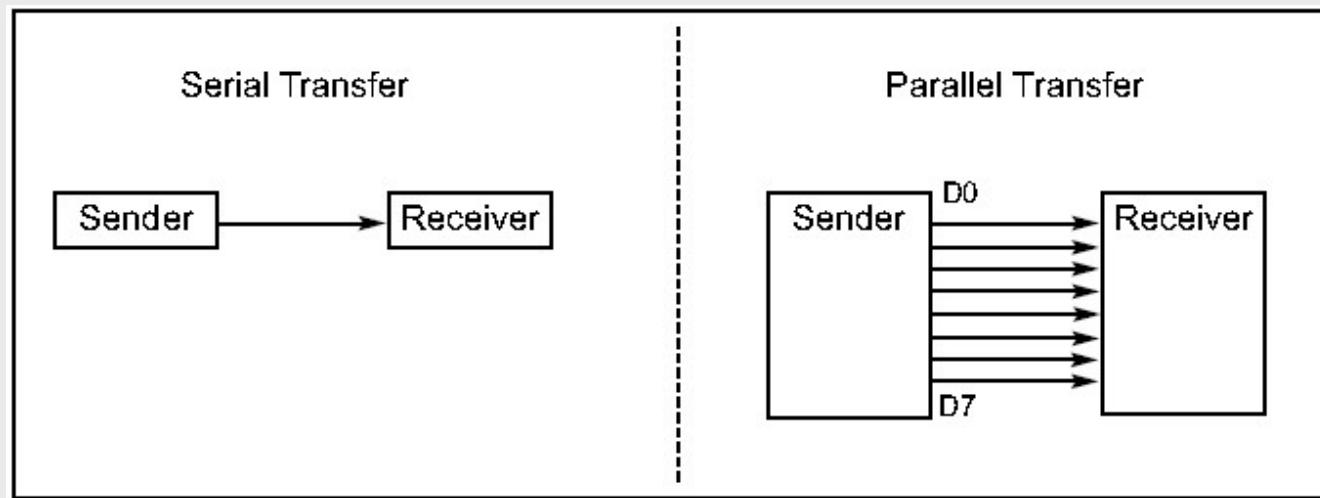
AVR Puerto Serial



Responde al a Norma RS232

Paralelo vs. Serie

- Paralelo
Transferencia de un byte de datos a la vez -> más rápido, más fácil
- Serial
Transfiere un bit después de otro -> más barato,
(se necesita un módem), ideales para largas distancias a través de la
línea telefónica



Direction

- Simple: los datos se mueve en una sola dirección
- Half Duplex: los datos se mueve en dos direcciones, pero no al mismo tiempo
- Full Duplex: los datos se mueve en dos direcciones al mismo tiempo

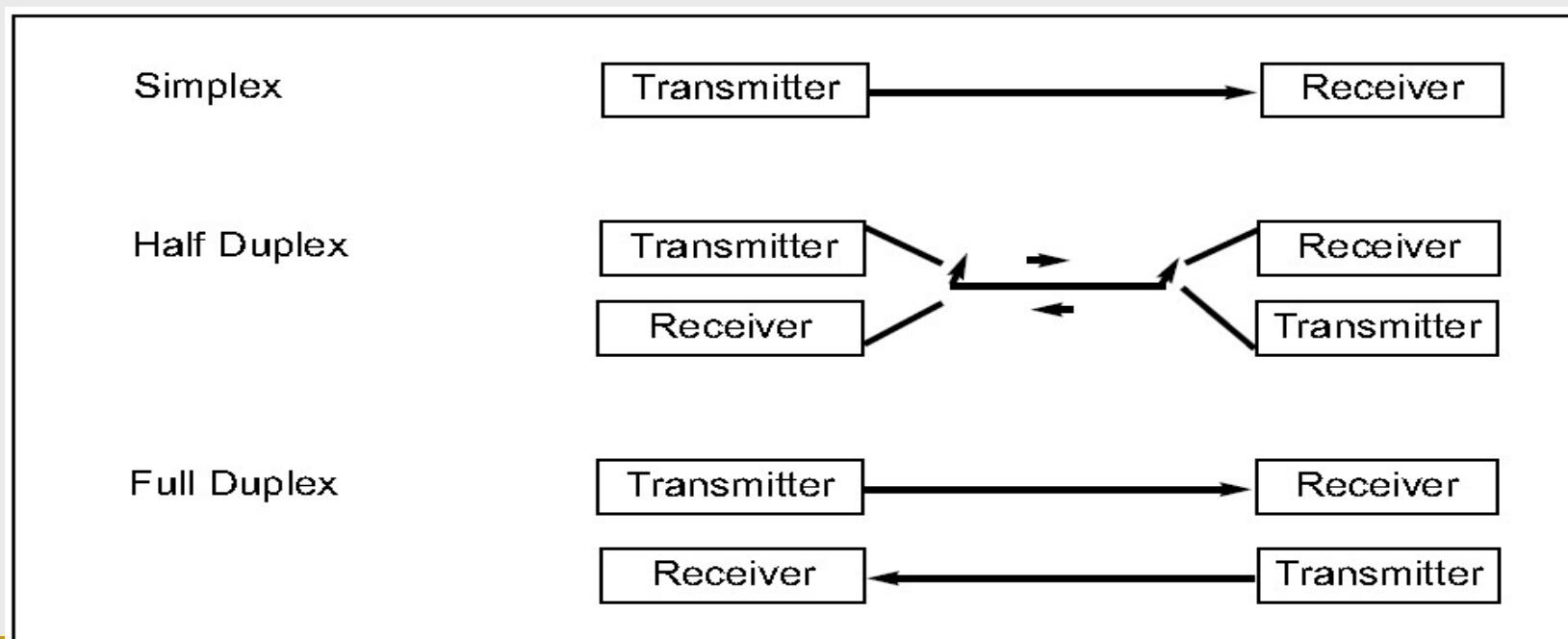


Figure 11-2. Simplex, Half-, and Full-Duplex Transfers

Síncronico vs. asíncronico

■ Síncronico

El clock debe ser transmitido durante la transmisión de datos.

Sólo un lado genera clock.

■ Asíncrono

□ Pulso de clock no se transmite.

Debe haber una manera de sincronizar las dos partes.

Data framing and bps

- Para sincronizar los dos lados, la trama que se utiliza: comienza con un espacio (0) que se llama Start
- Un carácter de 7-9 bits se transmite después de bit de inicio
- Cada trama se termina con uno
- El número de bits transmitidos en un segundo se llama bps

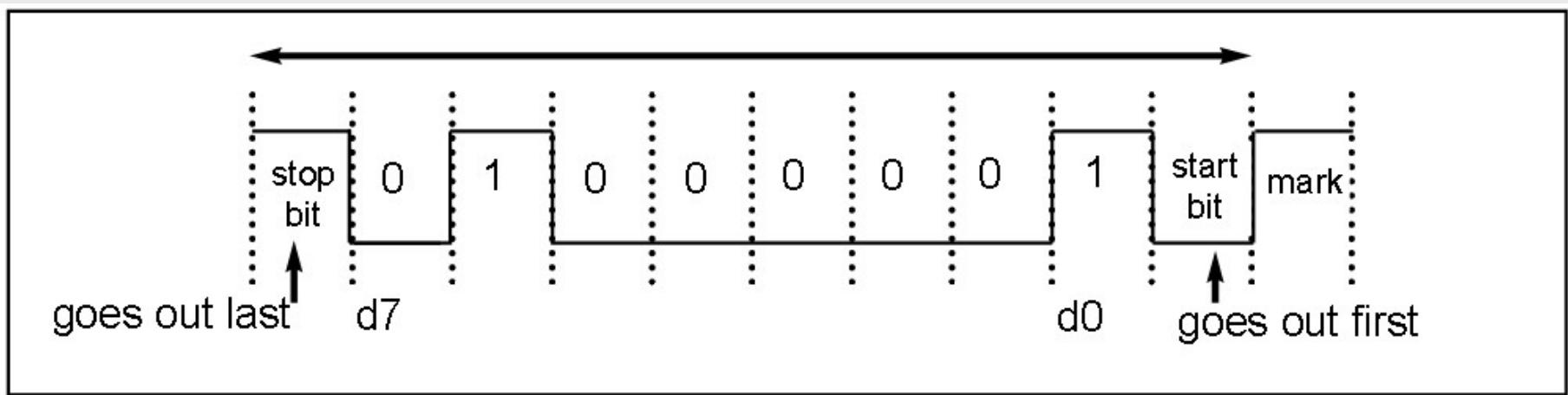


Figure 11-3. Framing ASCII 'A' (41H)

05/10/2018

15

AVR Connection

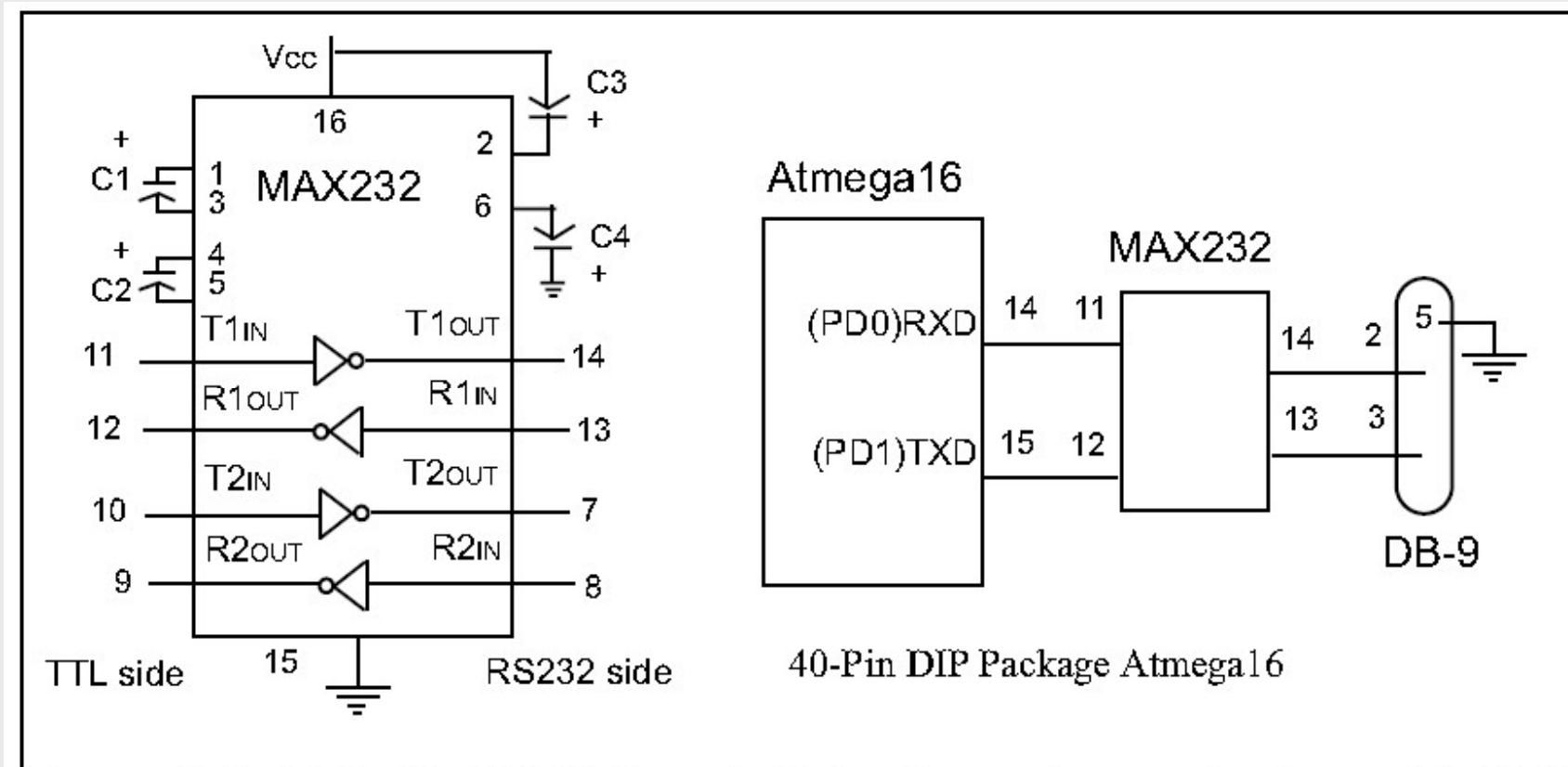
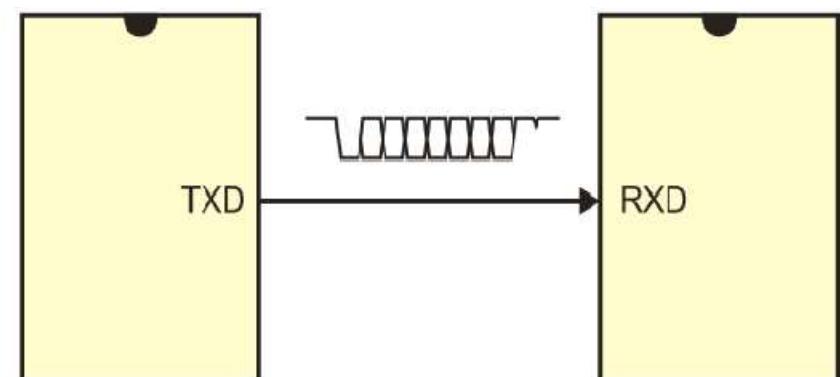
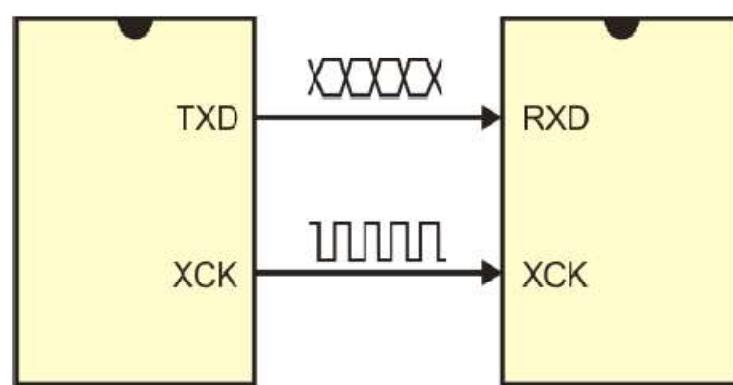


Figure 11-7. (a) Inside MAX232 and (b) its Connection to the Atmega16 (Null Modem)

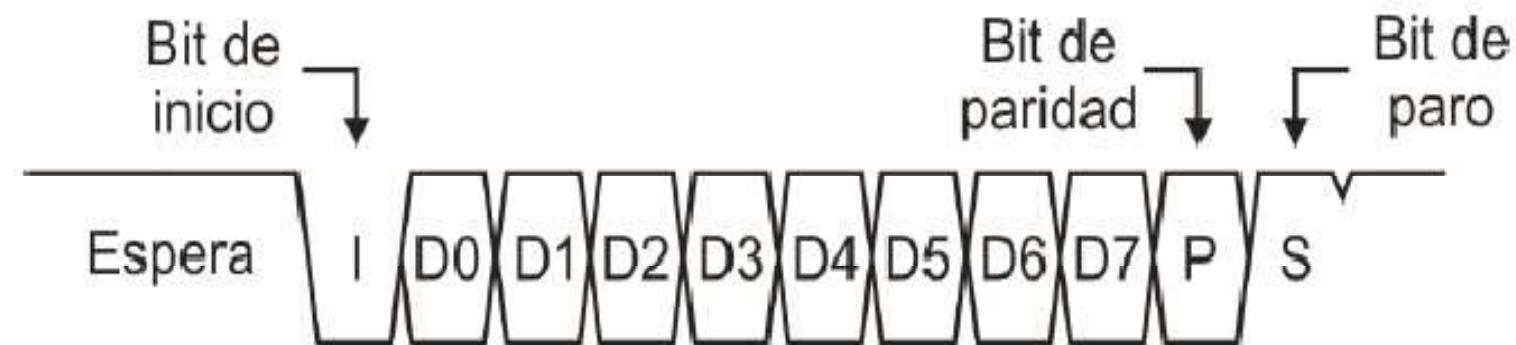
CARACTERÍSTICAS DE LOS USARTs

- **OPERACIÓN FULL DUPLEX, SINCRÓNICA Y ASINCRÓNICA**



CARACTERÍSTICAS DE LOS USARTs

- **SOPORTA FORMATOS DE 5, 6, 7, 8 Y 9 BITS CON 1 O 2 BITS DE PARADA**



- **GENERADOR DE PARIDAD PAR O IMPAR**
- **DETECCIÓN DE SOBREPOSICIÓN Y DE TRAMA**
- **TRES INTERRUPCIONES INDEPENDIENTES PARA LA RECEPCIÓN, TRANSMISIÓN Y DATOS VACÍO**

USART en AVR

- En AVR 5 registros están asociados con USART
 - UBRR (USART Baud Rate Register)
 - UDR (USART Data Register)
 - UCSRA (USART Control and Status Register A)
 - UCSRB (USART Control and Status Register B)
 - UCSRC (USART Control and Status Register C)

ORGANIZACIÓN DE LA USART

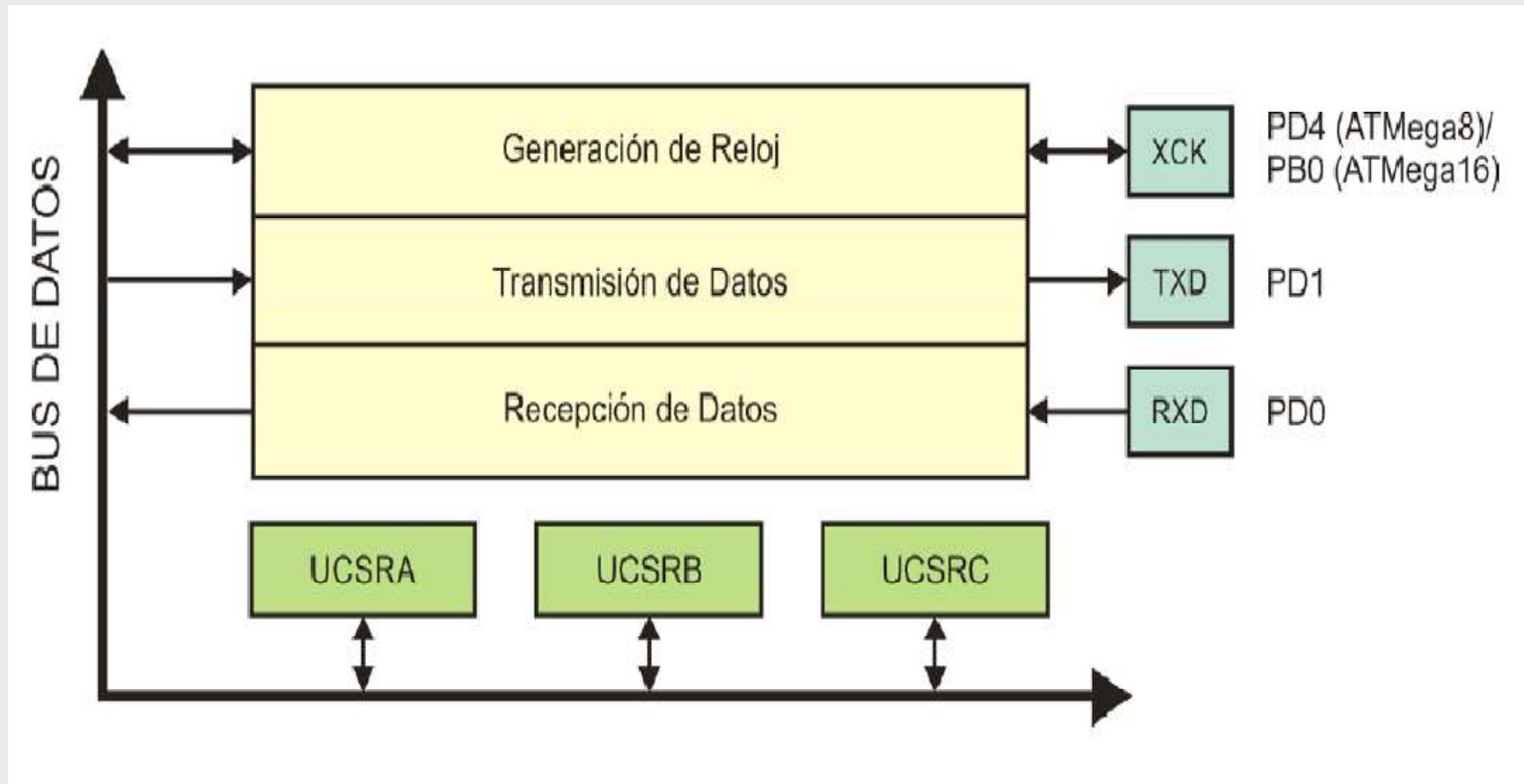


DIAGRAMA DE LOS USARTS

Out r18,UDRx

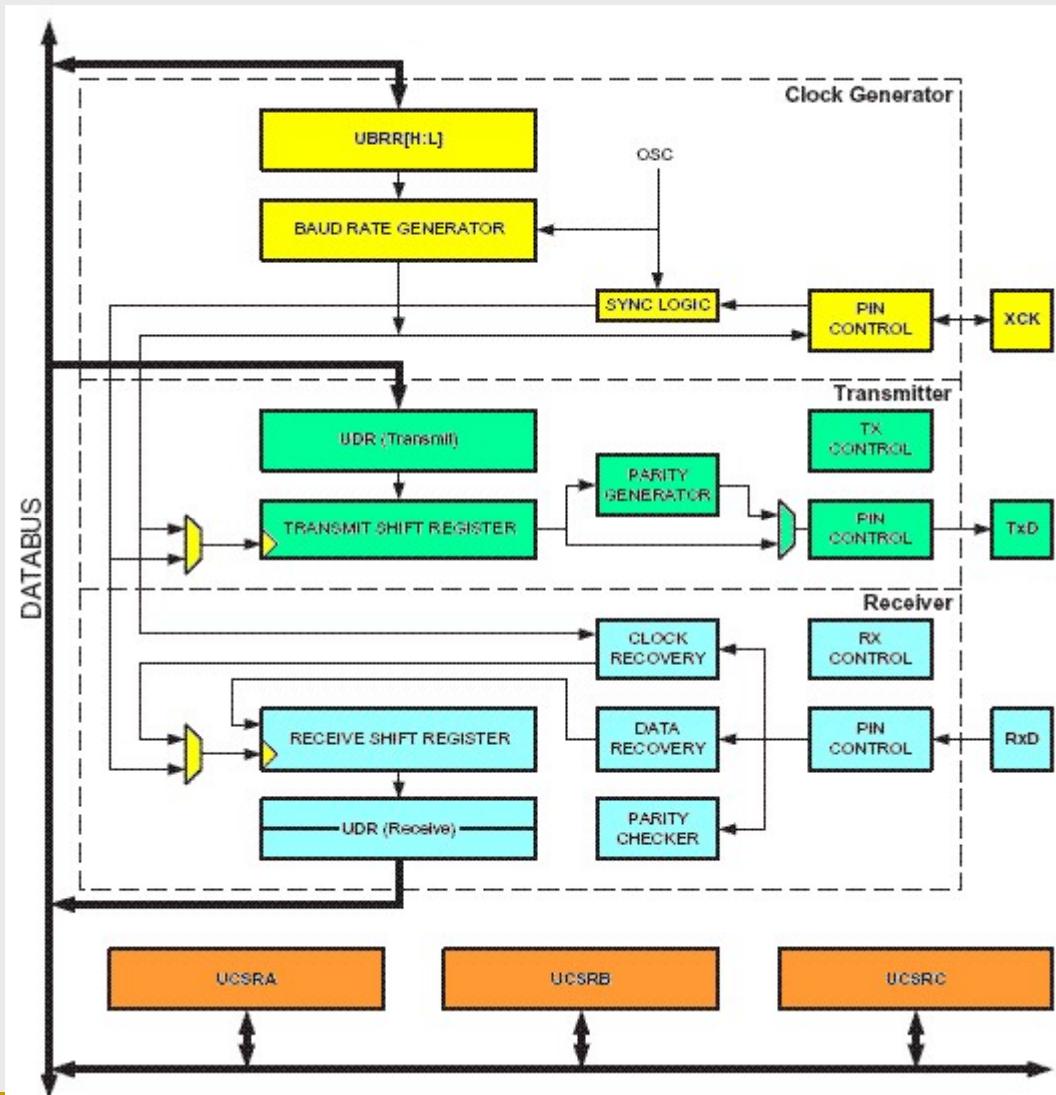
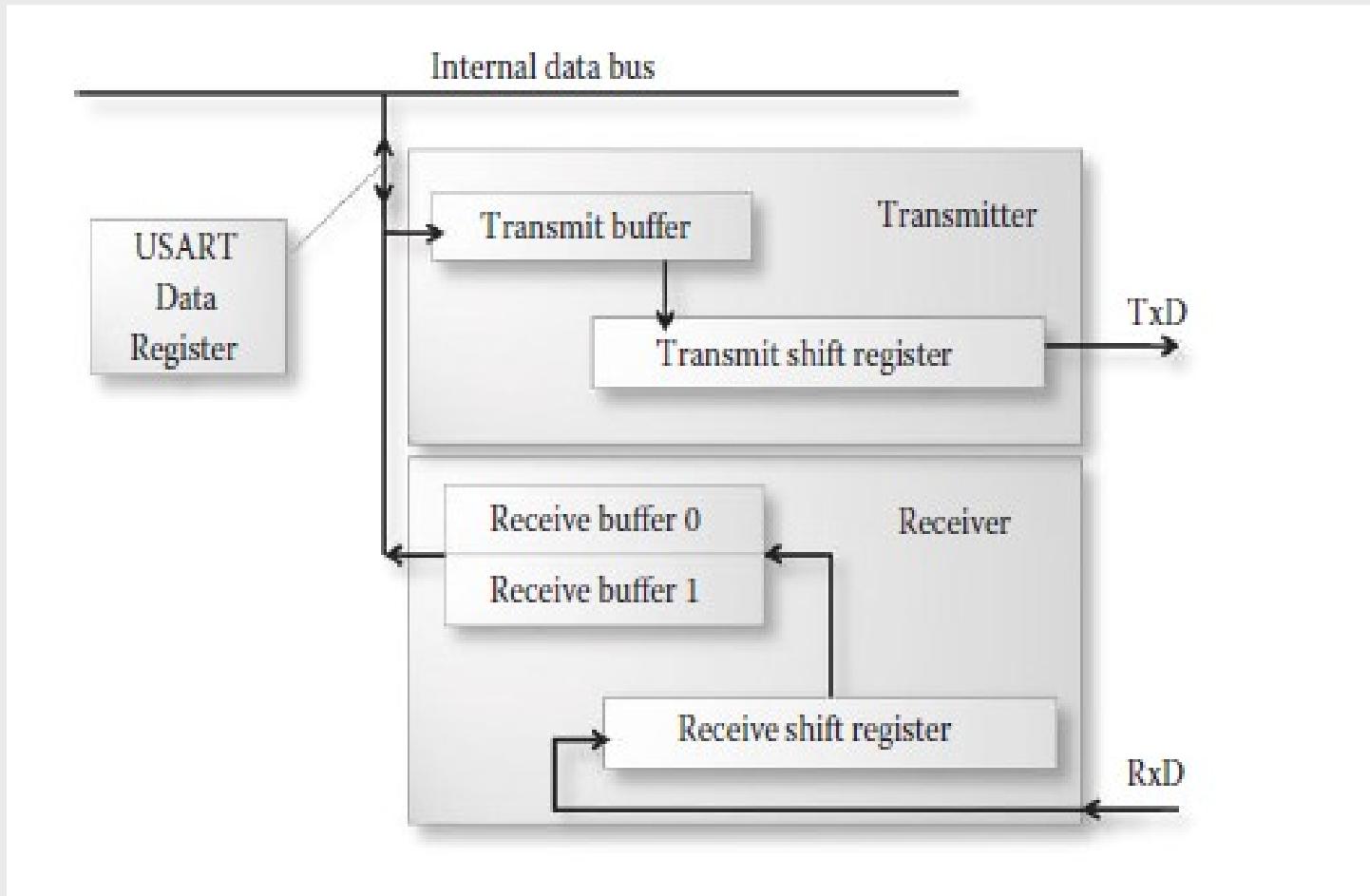
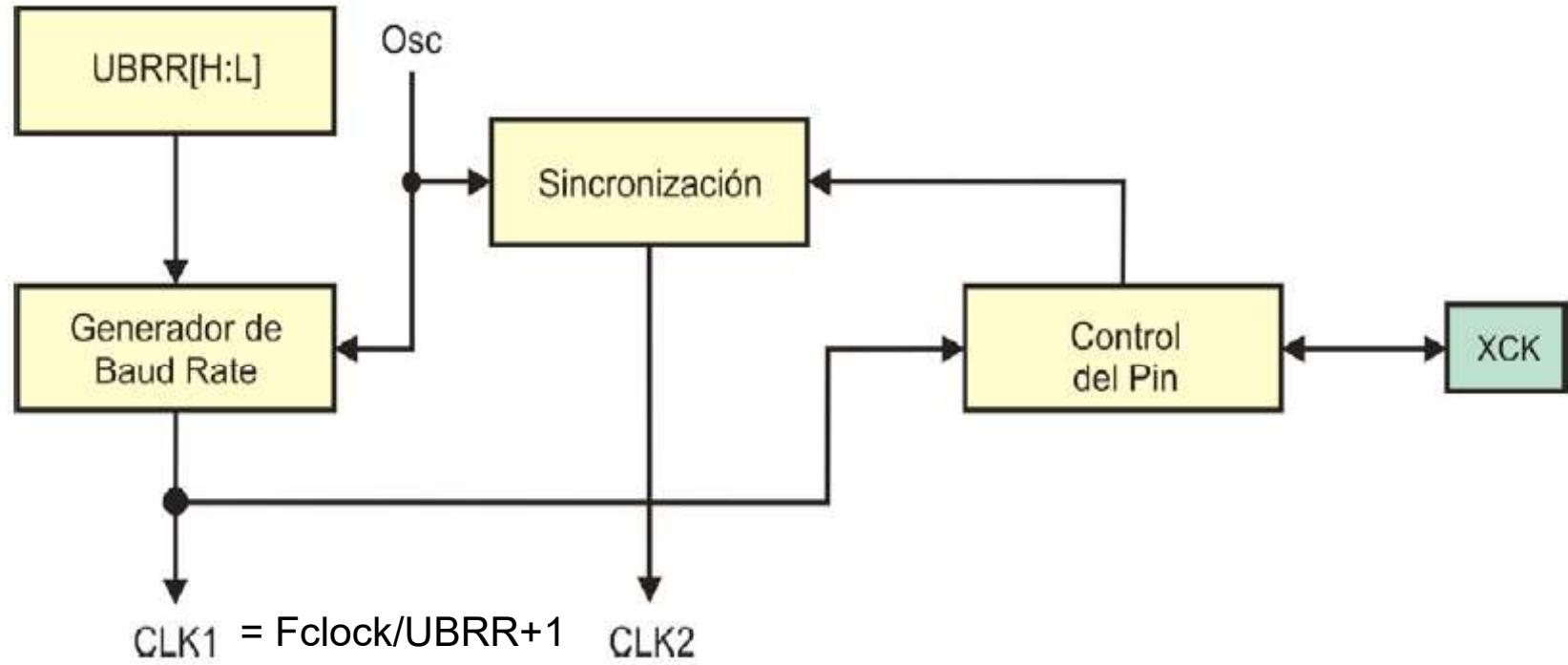


Diagrama en bloques USART simplificado

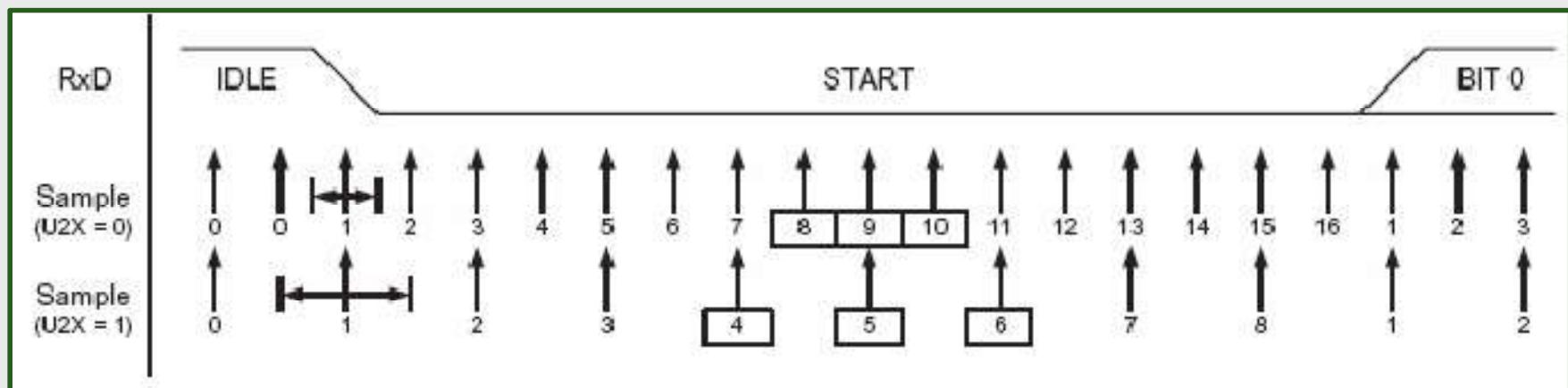
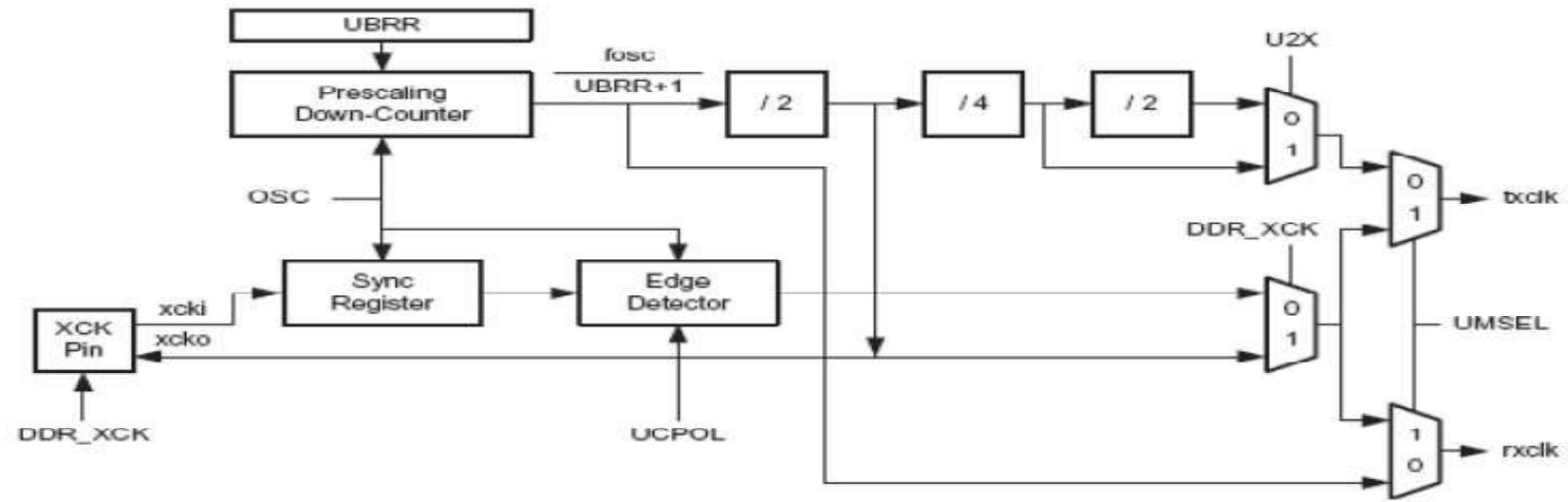


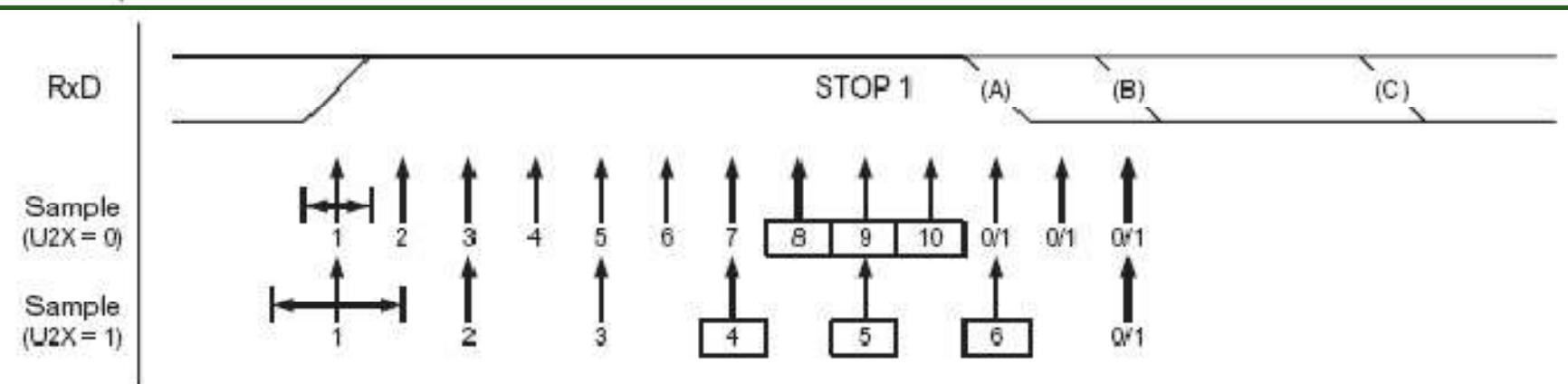
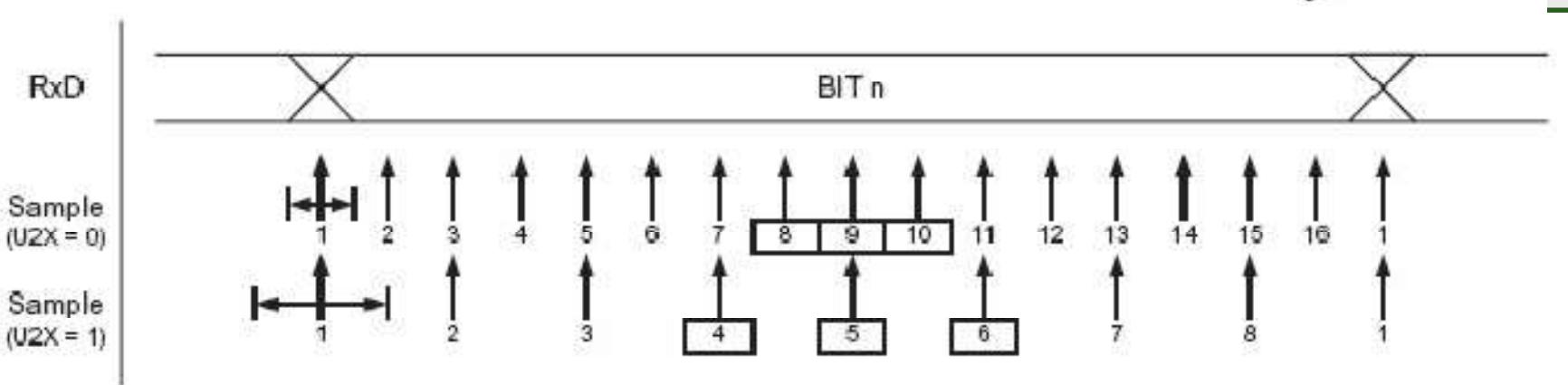
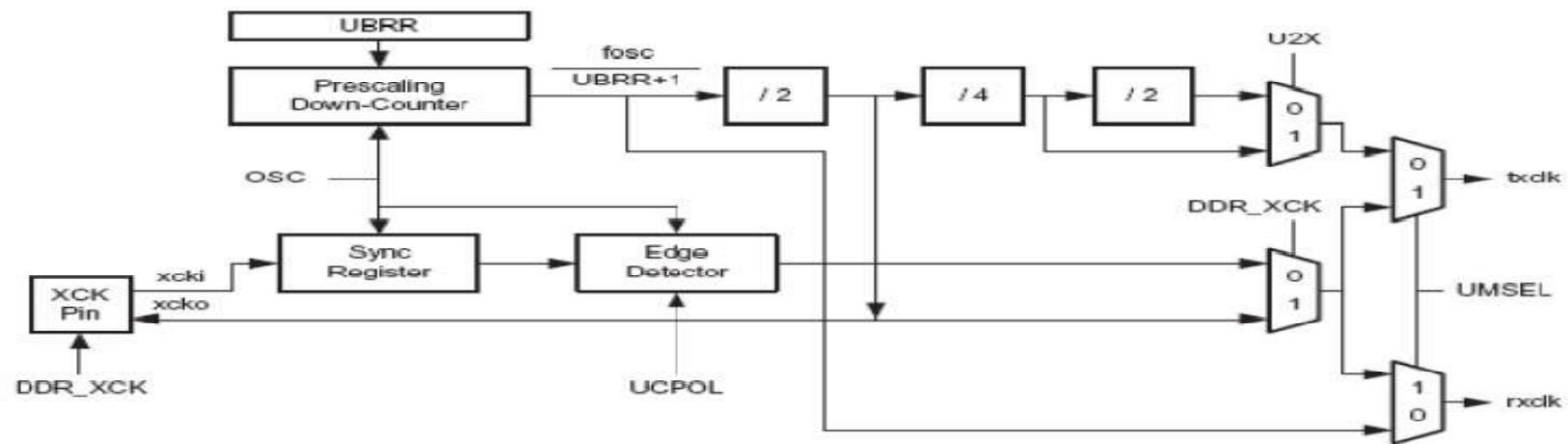
Bloque para la generación del reloj de la USART



- La señal **CLK1** es utilizado para **transmisión/recepción asíncrona** para transmisión síncrona
- La señal **CLK2** es el reloj utilizado para recepción **síncrona**.
- El registro **UBRR** es la base para la generación de la señal CLK1

Recuperación asincrónica de Datos





REGISTROS DE DATOS PARA E/S DE LOS USARTS

**REGISTROS DE
RECEPCIÓN, SOLO
DE LECTURA**

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UDR0				RXB0 [7:0]						(\$C6)
UDR1				RXB1 [7:0]						(\$CE)
UDR0				TXB0 [7:0]						(\$C6)
UDR1				TXB1 [7:0]						(\$CE)

❑ UDR (USART Data Register)

IN UDRx, R18

OUT R18, UDRX

**REGISTROS DE
TRANSMISIÓN, SOLO
DE ESCRITURA**

UDR es un puente entre usted y el registro de desplazamiento en serie
Si usted lee UDR, los datos se leen desde registro de desplazamiento recibido
Si se escribe a UDR, los datos se escriben en transmitir registro de desplazamiento

REGISTRO DE CONTROL “A” DEL USART **UCSR_A**

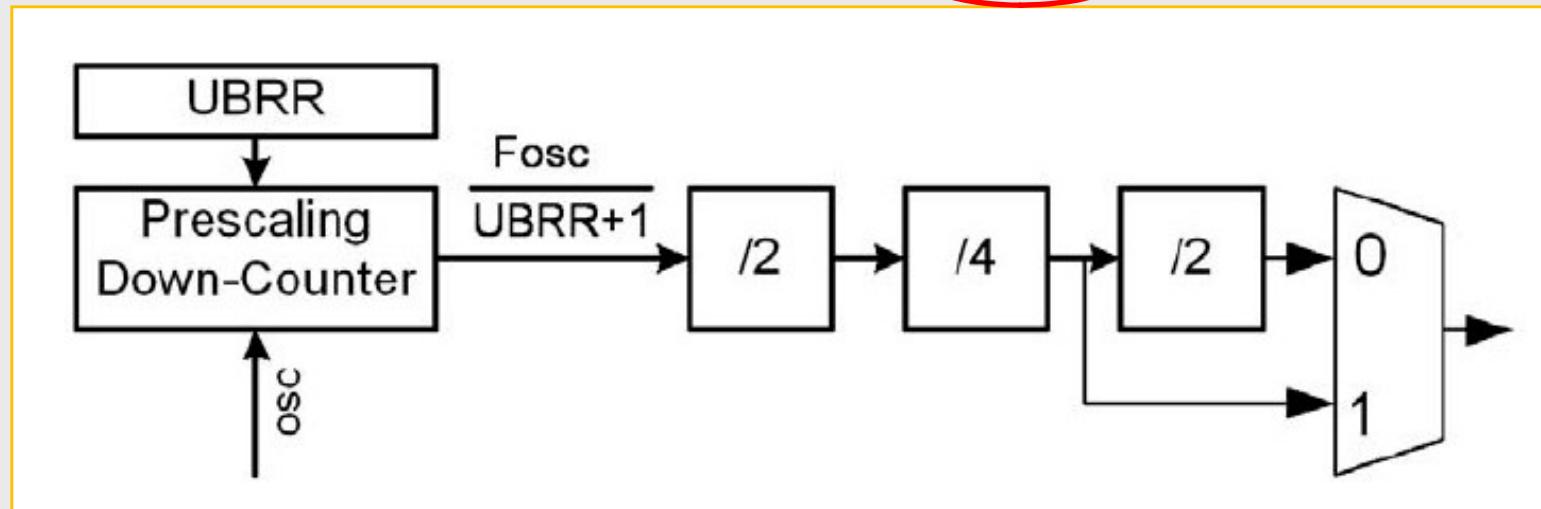
REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0		(\$C0)
UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1		(\$C8)
0x0B	7 RXC	6 TXC	5 UDRE	4 FE	3 DOR	2 PE	1 U2X	0 MPCM	UCSRA	

F	DESCRIPCIÓN
RXCn	Bandera de recepción completa
TXCn	Bandera de transmisión completa
UDREn	Bandera de registro de datos vacío
FEn	Bandera de error en la trama, bit de stop 0
DORn	Bandera de datos sobrepuestos
UPEn	Bandera de error de paridad
U2Xn	Duplicación de la velocidad
MPCMn	Modo de comunicaciones multiprocesadores

U2Xn

Duplicación de la velocidad

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0		(\$C0)
UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1		(\$C8)



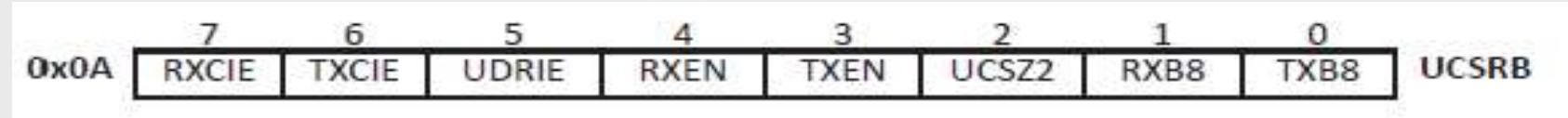
Dato	Paridad
0000 0001	0
0101 0001	0
0101 0101	1
0000 0000	1

Paridad Par UPE0/1

REGISTRO DE CONTROL “B” DEL USART

UCSR_B

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXENO	TXENO	UCSZ02	RXB80	TXB80		(\$C2)
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81		(\$C9)



DESCRIPCIÓN	
RXCIEn	Habilitación de la Interrupción en la Recepción
TXCIEn	Habilitación de la Interrupción en la Transmisión
UDRIEn	Habilitación de la Interrupción por datos vacío
RXENn	Habilitación de la recepción
TXENn	Habilitación de la transmisión
UCSZn2	Con UCSZ1:0 seleccionan el número de bits
RXB8n	Noveno bit de la recepción
TXB8n	Noveno bit de la transmisión

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXENO	TXENO	UCSZ02	RXB80	TXB80		(\$C2)
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81		(\$C9)

En el 9º bit

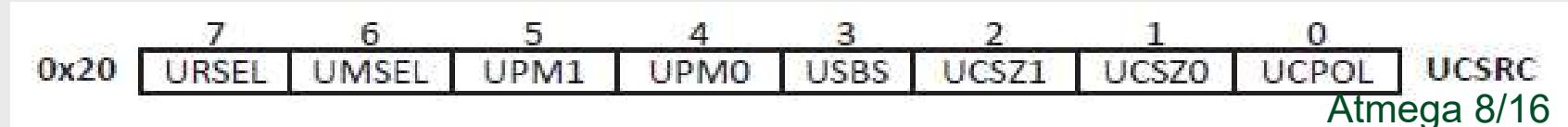
- Transición $(TXB8 \times) \rightarrow TB8$
- Recepción $(TB8) \rightarrow (RXB8x)$



REGISTRO DE CONTROL “C” DEL USART

UCSR_C

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOLO		(\$C2)
UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USBS1	UCSZ11	UCSZ01	UCPOL1		(\$CA)



Configuración	DESCRIPCIÓN
UMSELn1:0	Bits de selección del Modo Sincrónico, Asincrónico o Master SPI
UPMn1:0	Bits de selección de la paridad
USBSn	Bit de selección de los bits de parada
UCSZ1:0n	Para seleccionar el número de bits por carácter
UCPOLn	Bit de selección del flanco del reloj, solo en el modo sincrónico

SELECCIÓN DEL MODO DE OPERACIÓN

UMSELn1:0	DESCRIPCIÓN
00	USART Asincrónico
01	USART Sincrónico
10	Reservado
11	Master SPI, solo el USART1

REG.	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	I/O	SRAM
UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0		(\$C2)
UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USBS1	UCSZ11	UCSZ01	UCPOL1		(\$CA)

SELECCIÓN DE LA PARIDAD Y DE LOS BITS DE PARADA

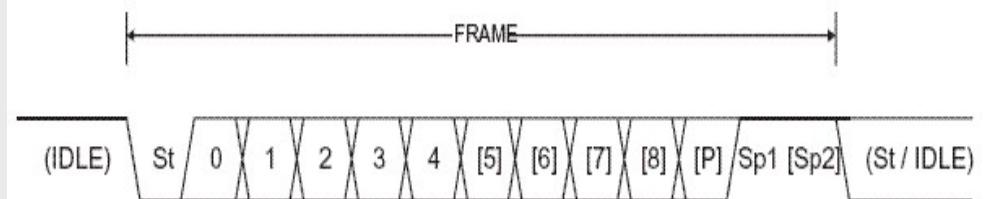
UPMn1:0	DESCRIPCIÓN
00	Deshabilitado
01	Reservado
10	Habilitado, paridad par
11	Habilitado, paridad impar

USBSn	DESCRIPCIÓN
0	1 bit de parada
1	2 bits de parada

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOLO		(\$C2)
UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USBS1	UCSZ11	UCSZ01	UCPOL1		(\$CA)

NÚMERO DE BITS POR CARACTER

UCSZn2:0	DESCRIPCIÓN
000	5 bits
001	6 bits
010	7 bits
011	8 bits
100	RESERVADO
101	RESERVADO
110	RESERVADO
111	9 bits

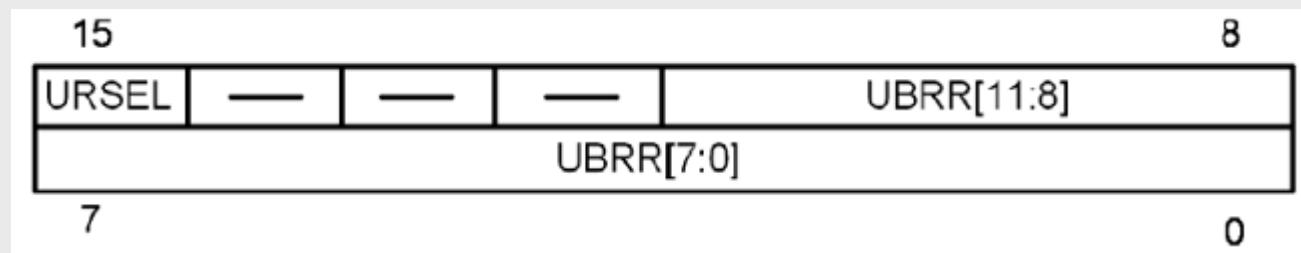


REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXENO	TXENO	UCSZ02	RXB80	TXB80		(\$C2)
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81		(\$C9)

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
0 UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USB\$0	UCSZ01	UCSZ00	UCPOLO		(\$C2)
UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USB\$1	UCSZ11	UCSZ01	UCPOL1		(\$CA)

Registro UBRR baud rate

- Para establecer la velocidad de transmisión en AVR debe establecer el valor de UBRR



$$\text{Desired Baud Rate} = \text{Fosc}/(16(\text{X} + 1))$$

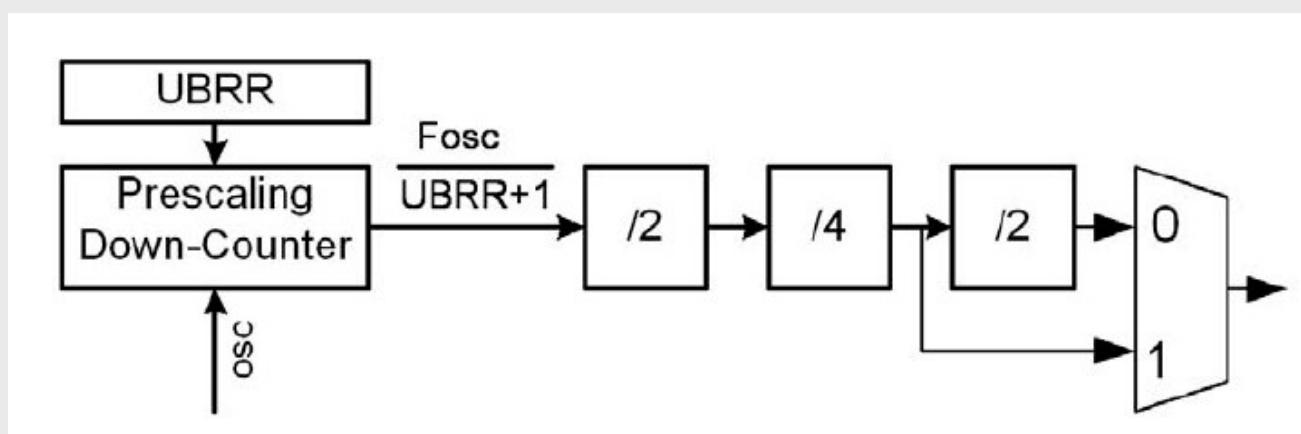


Table 11-3: Some PC Baud Rates in HyperTerminal

1,200
2,400
4,800
9,600
19,200
38,400
57,600
115,200

VELOCIDAD DE TRANSMISIÓN o BAUD RATE

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UBRR0H	-	-	-	-		UBRR0 [11:8]				(\$C4)
UBRR0L					UBRR0 [7:0]					(\$C5)
UBRR1H	-	-	-	-		UBRR0 [11:8]				(\$CC)
UBRR1L					UBRR0 [7:0]					(\$CD)

Operating Mode	Equation for Calculating Baud Rate	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Atmega 8/16

7	6	5	4	3	2	1	0		
0x20	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
7	6	5	4	3	2	1	0		
0x20	URSEL	-	-	-	UBRR11	UBRR10	UBRR9	UBRR8	UBRRH
0x09	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRRRL

El registro **UCSRC** comparte su dirección con **UBRRH** (0x20). El bit **URSEL** se utiliza para seleccionar el registro a escribir. Para leer a **UCSRC** deben realizarse 2 lecturas consecutivas, una lectura aislada proporciona el valor de **UBRRH**.

Con **URSEL** = 0 se escribe en **UBRRH** y con **URSEL** = 1 se escribe en **UCSRC**.

USART_ReadUCSRC:

; Read UCSR0C

in r16,UBRRH

in r16,UCSRC

ret

- La comunicación serial asíncrona puede funcionar a cualquier velocidad, siempre y cuando sea el mismo *baudrate* en ambas puntas.
- Pero: ningún dispositivo tiene la posibilidad de generar un clock a cualquier frecuencia arbitraria.
- Por eso existen ciertos baudrates estándar (9600, 19200, 38400, etc.).
- Pero: eso no garantiza que cualquier dispositivo pueda generar todos esos baudrates **exactos**, o con la misma exactitud unos que otros...
- Aunque el baudrate no sea exacto, hay un margen de diferencia admisible, dentro del cual el sistema todavía puede funcionar correctamente.
- Cada dispositivo tiene un porcentaje de error en la generación de cada baudrate. Podría darse el caso de que los errores se sumaran y la comunicación no fuera posible.

EJEMPLO NUMERICO.

Suponga que se desea configurar la UART a un Baud Rate de 19200. La frecuencia del oscilador (cristal) del microcontrolador es 4MHz.

$$(a) UBRR = \frac{4000000}{16(19200)} - 1 = 12.02$$

$$UBRR = \frac{f_{osc}}{16(BaudRate)} - 1$$

(b) Redondeando se tiene UBRR = 12

$$(c) Baud\ Rate_{REAL} = \frac{4000000}{16(12+1)} = 19230.769$$

$$\%Error = 100 \left| 1 - \frac{\text{Baud Rate}_{REAL}}{\text{Baud Rate}} \right|$$

$$(d) \%Error = 100 \left(1 - \frac{19230.769}{19200} \right) - 0.16 \cong 0.2$$

VALOR DE UBRR (1)

Baud Rate (bps)	$f_{osc} = 1.0000$ MHz				$f_{osc} = 1.8432$ MHz				$f_{osc} = 2.0000$ MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	-	-	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	-	-	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	-	-	-	-	-	-	0	0.0%	-	-	-	-
250k	-	-	-	-	-	-	-	-	-	-	0	0.0%
Max ⁽¹⁾	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

VALOR DE UBRR (2)

Baud Rate (bps)	$f_{osc} = 3.6864$ MHz				$f_{osc} = 4.0000$ MHz				$f_{osc} = 7.3728$ MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	—	—	0	-7.8%	—	—	0	0.0%	0	-7.8%	1	-7.8%
1M	—	—	—	—	—	—	—	—	—	—	0	-7.8%
Max ⁽¹⁾	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

EJEMPLO

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSROC	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOLO		(\$C2)
UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USBS1	UCSZ11	UCSZ01	UCPOL1		(\$CA)

;PASO 1

- **cli ;SREG<|>=0** ;int. deshabilitadas. ■ ;PASO 2
- **cbi DDRD,0** ;Pin RXD como entrada
- **cbi DDRD,1** ;Pin TXD como salida ■ ;PASO 3
- **ldi r17,0x0c** ;Baud Rate=19200 ■ ;PASO 4
- **out UBRR,r17;**
- **cbi UCSR0B,UCSZ02** ;Longitud de Datos 8 bits
- **Sbi UCSR0B,UCSZ01** ;011
- **sbi UCSR0B,UCSZ00**
- ;PASO 5
- **cbi UCSR0B,RXCIE** ;Interrupción de Recepción. NO HABILITADA.
- **cbi UCSR0B,TXCIE** ;Interrupción de Transmisión. NO HABILITADA.
- **Cbi UCSR0B,UDRIE** ;Interrupción de UDR vacío .No Habilitada.

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXENO	TXENO	UCSZ02	RXB80	TXB80		(\$C2)
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81		(\$C9)

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXENO	TXENO	UCSZ02	RXB80	TXB80		(\$C2)
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81		(\$C9)

- ;PASO 6
 - sbi UCSR1B,TXEN0 ;Habilita Transmisión
 - Sbi UCSR1B,RXEN0 ;Habilita Recepción.

- ;PASO 7
 - sei ;Habilita Interrupciones Globales
 - loop:
 - nop
 - nop
 - Sbis UCSR1A,RXC0 ; ¿Llego nuevo Dato?
 - rjmp loop ;No, seguir esperando
 - in r18,UDR0 ;Si, guardararlo en r18
 - out UDR0,r18 ;Hacer eco.
 - nop
 - nop
 - rjmp loop ;loop infinito

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0		(\$C0)
UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1		(\$C8)

Program the ATmega32 to receive bytes of data serially and put them on Port B. Set the baud rate at 9600, 8-bit data, and 1 stop bit. Use Receive Complete Interrupt instead of the polling method.

Solution:

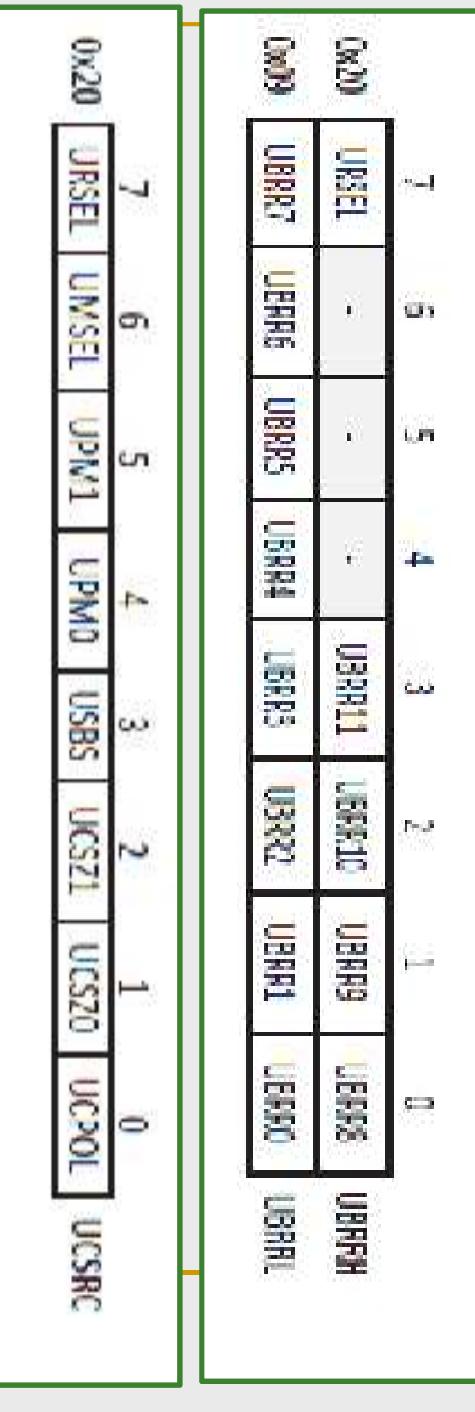
```
.INCLUDE "M32DEF.INC"
.CSEG
    RJMP MAIN
.ORG URXCaddr
    RJMP URXC_INT_HANDLER
.ORG 40

MAIN: LDI R16, HIGH(RAMEND)
      OUT SPH, R16
      LDI R16, LOW(RAMEND)
      OUT SPL, R16
      LDI R16, (1<<RXEN) | (1<<RXCIE)
      OUT UCSRB, R16
      LDI R16, (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL);sync,8-bit data
      OUT UCSRC, R16
      LDI R16, 0x33
      OUT UBRRL, R16
      LDI R16, 0xFF
      OUT DDRB, R16
      SEI

WAIT_HERE:
    RJMP WAIT_HERE
    ;stay here

URXC_INT_HANDLER:
    IN R17, UDR
    OUT PORTB, R17
    RETI
      ;send UDR to R17
      ;send R17 to PORTB

      ;put in code segment
      ;jump main after reset
      ;int-vector of URXC int.
      ;jump to URXC_INT_HANDLER
      ;start main after
      ;interrupt vector
      ;initialize high byte of
      ;stack pointer
      ;initialize low byte of
      ;stack pointer
      ;enable receiver
      ;and RXC interrupt
      ;no parity, 1 stop bit
      ;9600 baud rate
      ;set Port B as an
      ;input
      ;enable interrupts
```



Write a program for the AVR to transmit the letter 'G' serially at 9600 baud, continuously. Assume XTAL = 8 MHz. Use interrupts instead of the polling method.

Solution:

```

INCLUDE "M32DEF.INC"

.CSEG
    RJMP MAIN
.ORG UDREaddr
    RJMP UDRE_INT_HANDLER
.ORG 40

;*****
MAIN:
    LDI R16, HIGH(RAMEND)           ;initialize high byte of
    OUT SPH, R16                   ;stack pointer
    LDI R16, LOW(RAMEND)           ;initialize low byte of
    OUT SPL,R16                   ;stack pointer
    LDI R16, (1<<TXEN) | (1<<UDRIE) ;enable transmitter
    OUT UCSR B, R16                ;and UDRE interrupt
    LDI R16, (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL); sync., 8-bit
    OUT UCSR C, R16                ;data no parity, 1 stop bit
    LDI R16, 0x33                  ;9600 baud rate
    OUT UBRR L, R16
    SEI                           ;enable interrupts

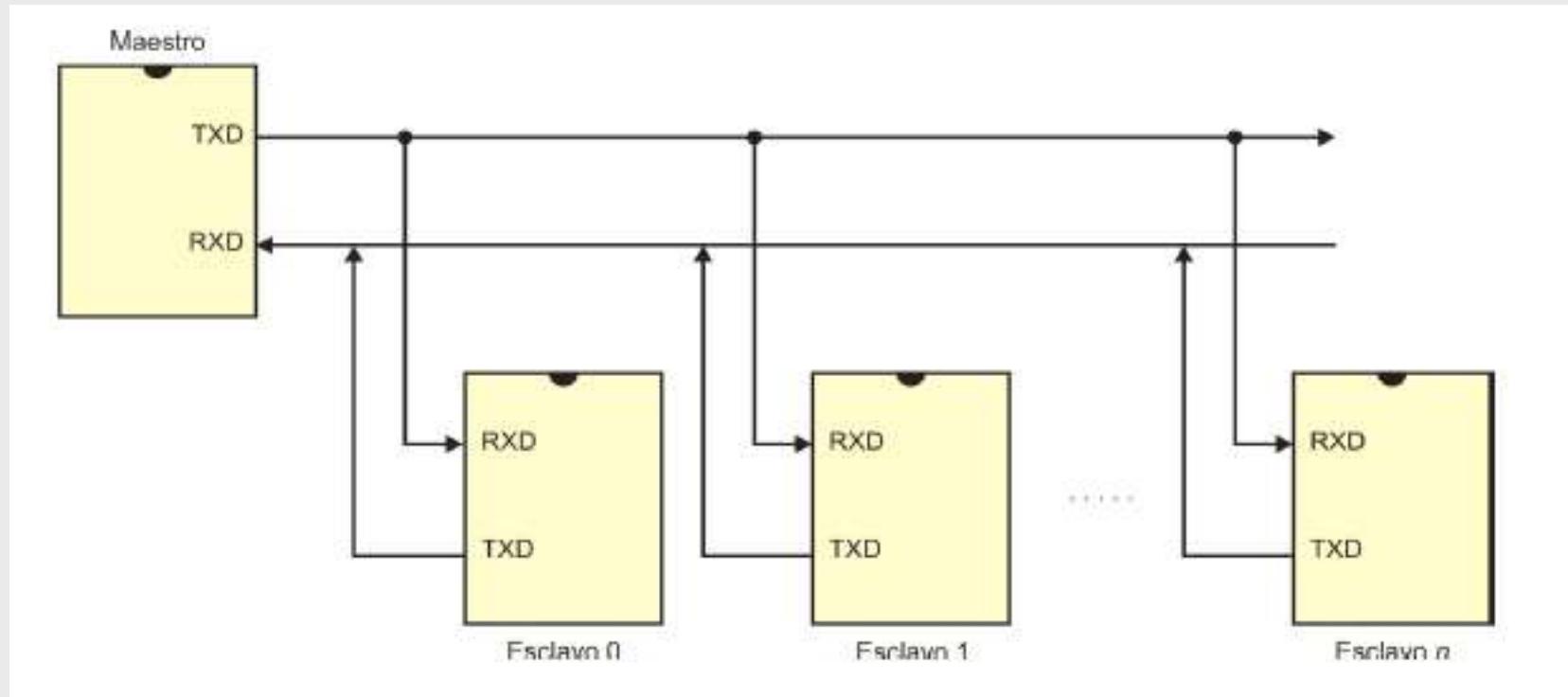
WAIT_HERE:
    RJMP WAIT_HERE                 ;stay here
;*****

UDRE_INT_HANDLER:
    LDI R26, 'G'                   ;send 'G'
    OUT UDR, R26                   ;to UDR
    RETI

```

UDRE0 Este bit al ponerse a 1 en forma automática indica que el registro UDR0 está vacío por lo que se le podrá cargar con algún dato. Cuando se cargue con algún valor el registro UDR0 este bit se pondrá automáticamente a 0. Se puede habilitar la interrupción por detección de que el registro UDCR0 está vacío y este bit será el que indique esa interrupción

COMUNICACIÓN ENTRE MICROCONTROLADORES



La comunicación utiliza un formato de 9 bits, donde el 9º bit (transmitido en **TXB8** o recibido en **RXB8**) sirve para distinguir entre dos tipos de información, con 0 se refiere a una trama de datos y con 1 a una trama de dirección.

COMUNICACIÓN ENTRE MICROCONTROLADORES

Este modo implica un esquema maestro-esclavos. Cada esclavo tendrá una dirección que lo distinga de los demás. Pueden ser hasta 256 esclavos.

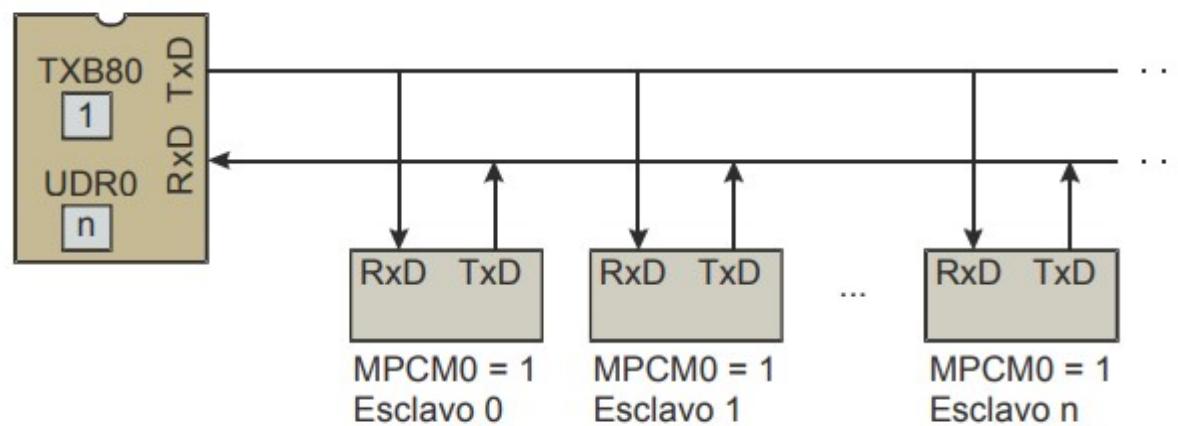
La comunicación se realiza con un formato de 9 bits, donde el 9º bit (transmitido en TXB80 o recibido en RXB80) sirve para distinguir entre campos de datos o dirección:

0	Datos
1	Dirección

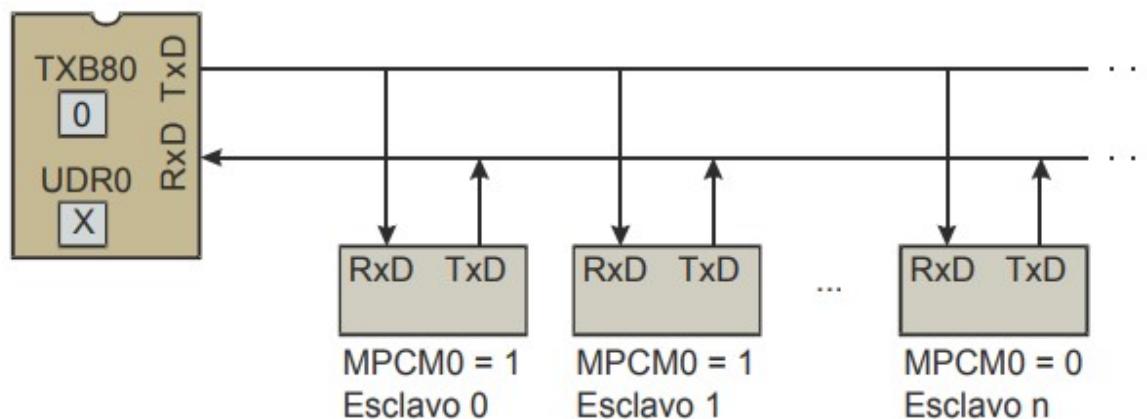
REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0		(\$C0)
UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1		(\$C8)

Se realiza de la siguiente manera:

1. Todos los esclavos deben habilitar el modo de multiprocesadores, con $MPCM0 = 1$. Con ello únicamente pueden recibir campos de dirección.
2. El maestro envía una dirección, la cual es recibida por todos los esclavos.



3. Cada esclavo debe leer su registro UDR0 para determinar si ha sido seleccionado. El que resulte seleccionado debe limpiar su bit MPCM0 en el registro UCSR0A.



4. El esclavo seleccionado y el maestro pueden realizar un intercambio bidireccional de datos, el cual pasa desapercibido por los demás esclavos, por que aún tienen su bit MPCM en alto.
5. Cuando se concluye el diálogo, el esclavo seleccionado debe poner en alto su bit MPCM, quedando como todos los esclavos, en espera de que el maestro solicite su atención.

COMUNICACIÓN ENTRE MICROCONTROLADORES

1. Los esclavos habilitan el modo de comunicación entre multiprocesadores, con **MPCM = 1**. De manera que únicamente pueden recibir tramas de dirección (9º bit en 1).
2. El maestro envía la dirección del esclavo con el que va a interactuar, esta dirección es recibida por todos los esclavos.
3. Cada esclavo lee su registro **UDR** y lo compara con su dirección para determinar si ha sido seleccionado. El esclavo seleccionado limpia su bit **MPCM** en el registro **UCSRA**, quedando disponible para recibir datos.
4. El maestro y el esclavo seleccionado realizan el intercambio de datos (9º bit en 0), el cual pasa desapercibido por los otros esclavos, porque aún tienen su bit **MPCM** en alto.
5. Cuando el diálogo concluye, el esclavo seleccionado debe poner en alto su bit **MPCM**, quedando como los demás esclavos, en espera de que el maestro solicite su atención.

TEMPORIZADORES \

TIMER- AVR



Temporizadores \ Timers- AVR

LOS TIMERS SON REGISTROS CONTADORES ASCENDENTES
LOS ATmega POSEEN EN GRAL. TRES TIMERS: TIMER 0, TIMER 1 Y TIMER 2
SE CONFIGURAN MEDIANTE LOS BITS DE LOS REGISTROS DE CONTROL

1. • Temporizadores / contadores de 8 bits
2. • Temporizadores / contadores de 16 bits

Características

1. • Prescaler reloj de entrada
2. • Estado del contador de lectura / escritura
3. • Generador de forma de onda utilizando un comparador (registro)
4. • Generador de PWM (modulación de ancho de pulso)
5. • Generación de interrupciones a intervalos de tiempo regulares.
6. • Activado por eventos externos (captura)

Usos

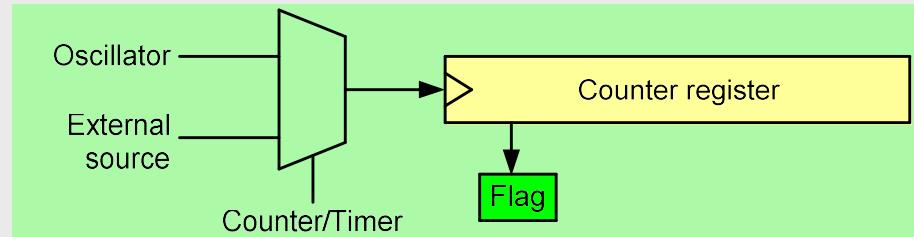
1. • Generador de forma de onda
2. • Programa de sincronización con intervalos de tiempo regulares.
3. • Medición de intervalos de tiempo.

Timer 0 en AVR

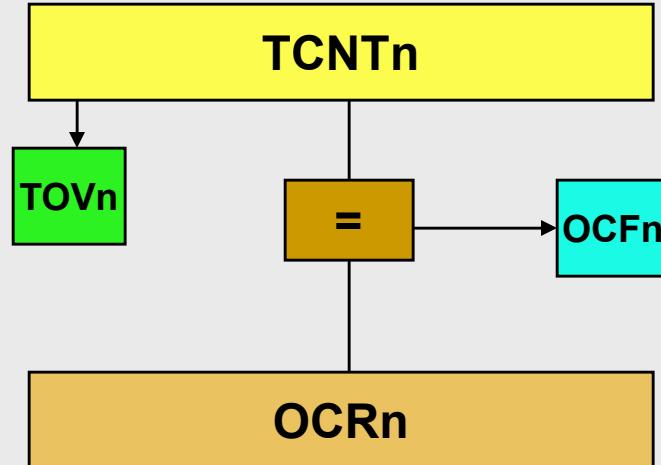
- **TCNT_n** (Timer/Counter register)
- **TOV_n** (Timer Overflow flag)
- **TCCR_n** (Timer Counter control register)
- **OCR_n** (output compare register)
- **OCF_n** (output compare match flag)
- **TIMSK** (Reg. E modo interrupción)
- **TIFR**

Comment:

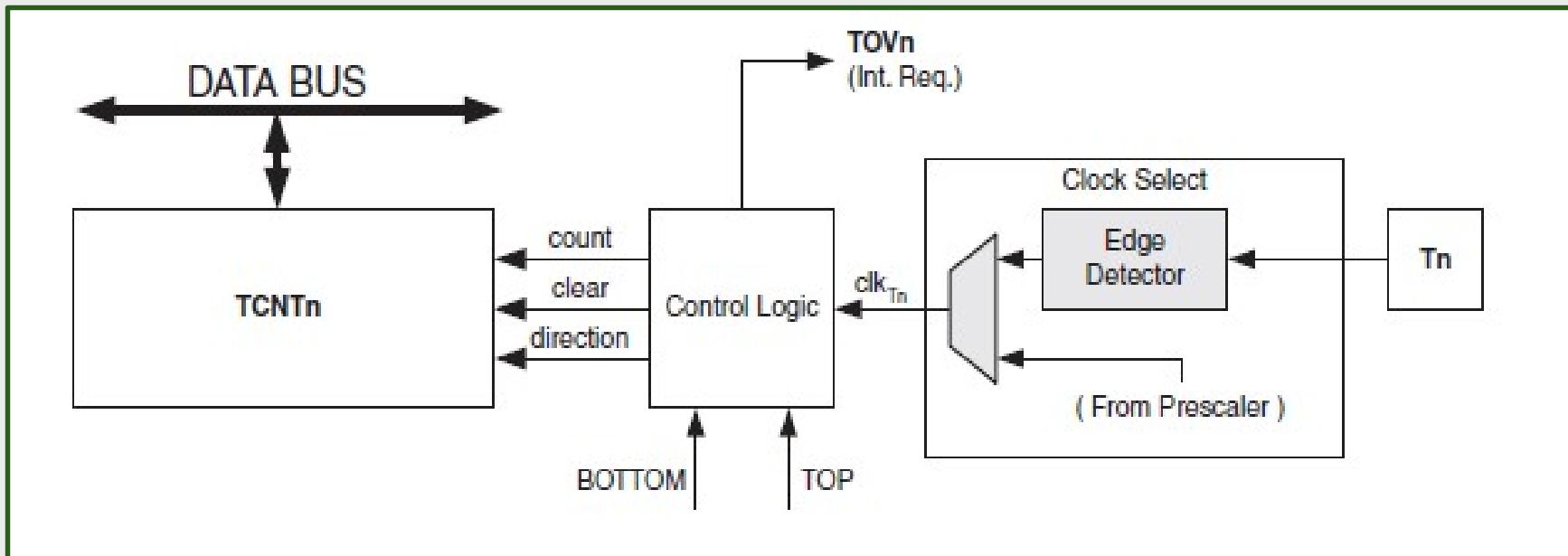
Todos los registros son byte-addressable I/O registers



TCCR_n



TIMER 0



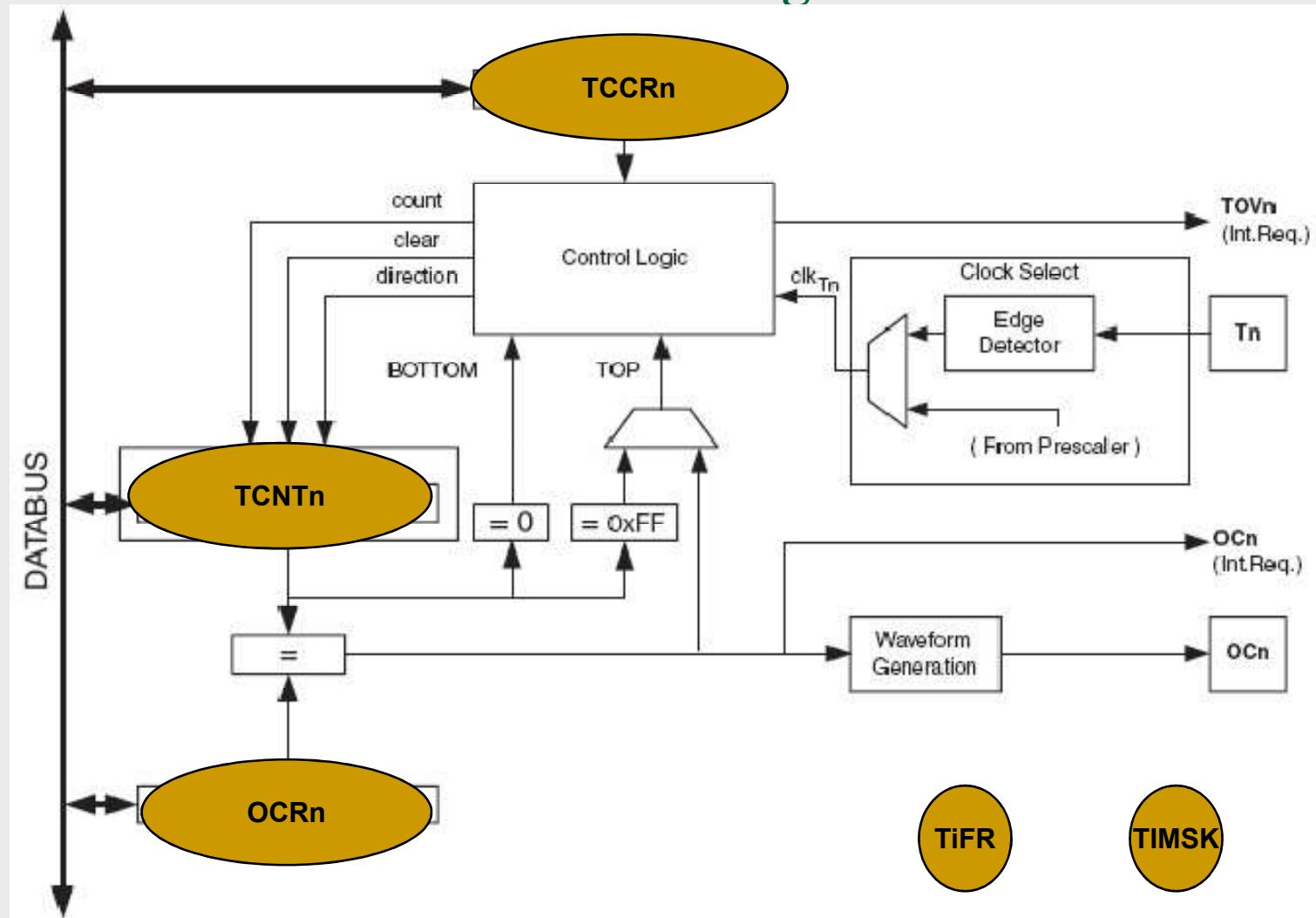
- **Count** Incremento o decremento TCNT0 en 1.
- **Dirección** Selecciona entre incremento y decremento.
- **Clear** Borrar TCNT0 (establecer todos los bits a cero).
- **ClkT0** Timer / contador de reloj.
- **Top** Señales en que TCNT0 ha alcanzado el valor máximo (0xFF).
- **Botton** TCNT0 ha alcanzado el valor mínimo (cero).
- El indicador de desbordamiento del contador / temporizador (TOV0) se configura de acuerdo con el modo de operación seleccionado por el WGM02

Registros

- TCCR0A** •**TCCR2A** •**Timer/Counter Control Register A**
- TCCR0B** •**TCCR2B** •**Timer/Counter Control Register B**
- TIMSK0** •**TIMSK2** •**Timer/Counter Interrupt Mask Register**
- TIFR0** •**TIFR2** •**Timer/Counter Interrupt Flag Register**

TCNT0	TCNT2	Timer/Counter Register
OCR0A	OCR2A	Output Compare Register A
OCR0B	OCR2B	Output Compare Register B

Diagrama simplificado de bloques Timer 0 ATmega32



TIMER 0

AVR ATMEGA 16

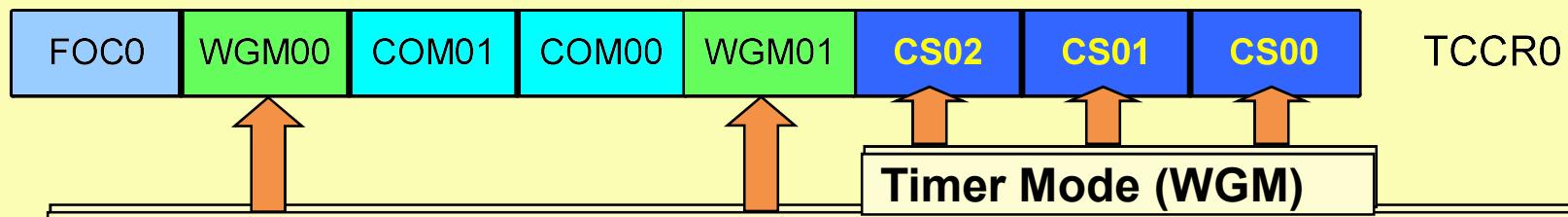
Timer Counter CONTROL Register

FO0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
0	0	0						Timer stop
0	0	1						clk No Prescale
0	1	0						clk/8
0	1	1						clk/64
1	0	1						clk/1024
1	1	0						Count on T0 fall
1	1	1						Count on T0 Rising edge

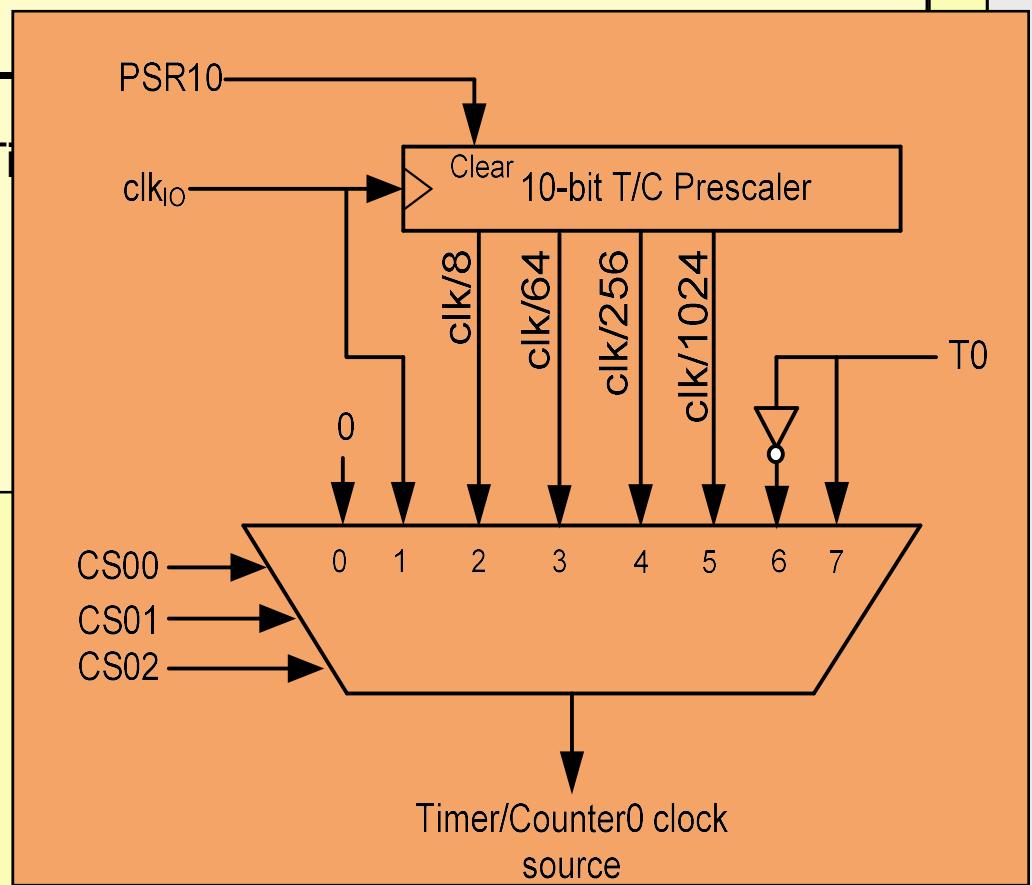


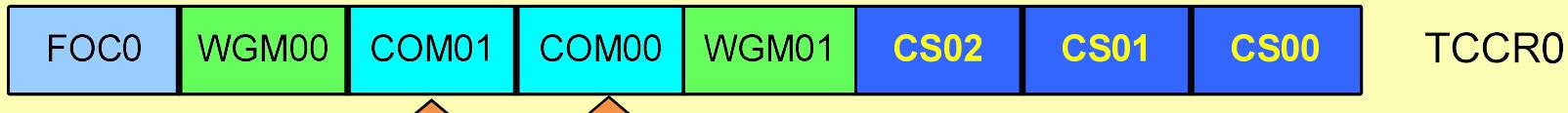
TOIE - TIMER OVERFLOW Interrupt Enable
OCIE - Output Compare Interrupt Enable

TOIE0 , OCIE0
TIMER 0 Interrupts



WGM00	WGM01	Comment
0	0	Normal
0	1	CTC (Clear Timer On Compare)
1	0	PWM, phase correct
1	1	Fast PWM





Compare Output Mode (COM)

CTC or Normal
(Non PWM)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Fast PWM

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP
1	1	Set OC0 on compare match, clear OC0 at TOP

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 73 for more details.

Phase Correct
PWM

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

TIMSK: (Timer Interrupt Mask Register)

Bit	7	6	5	4	3	2	1	0	
Read/Write	R/W	TIMSK							
Initial Value	0	0	0	0	0	0	0	0	

- **TOIE0:** Escribir a 1 en este bit habilita la interrupción por desbordamiento del TimerCounter0 (Timer/Counter Overflow Interrupt Enable 0) si también se habilitan las interrupciones globales con el bit I en SREG.
- **OCIE0:** Escribir a 1 en este bit habilita la interrupción por coincidencia en la comparación del Timer/Counter0 con el registro OCR0 (Output Compare Interrupt Enable 0), si también se habilitan las interrupciones globales con el bit I en SREG.

TIFR: TIMER INTERRUPT FLAG REGISTER

Bit	7	6	5	4	3	2	1	0	TIFR
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **TOV0:** Cuando este bit se pone a 1 se genera la interrupción por desbordamiento del Timer/Counter0. Este bit se limpia cuando entra a la función que atiende la interrupción o escribiendo un 1 en este bit.
- **OCF0:** Cuando este bit se pone a 1 se genera la interrupción por coincidencia en la comparación del Timer/Counter0 con el registro OCR0. Este bit se limpia cuando entra a la función que atiende la interrupción o escribiendo un 1 en este bit.

Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	TCNT0
TCNT0[7:0]									
ReadWrite	R/W	TCNT0							
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0 Register.

Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0	OCR0
OCR0[7:0]									
ReadWrite	R/W	OCR0							
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

Timer/Counter

Diagrama de bloque timer 0 ATmega164P

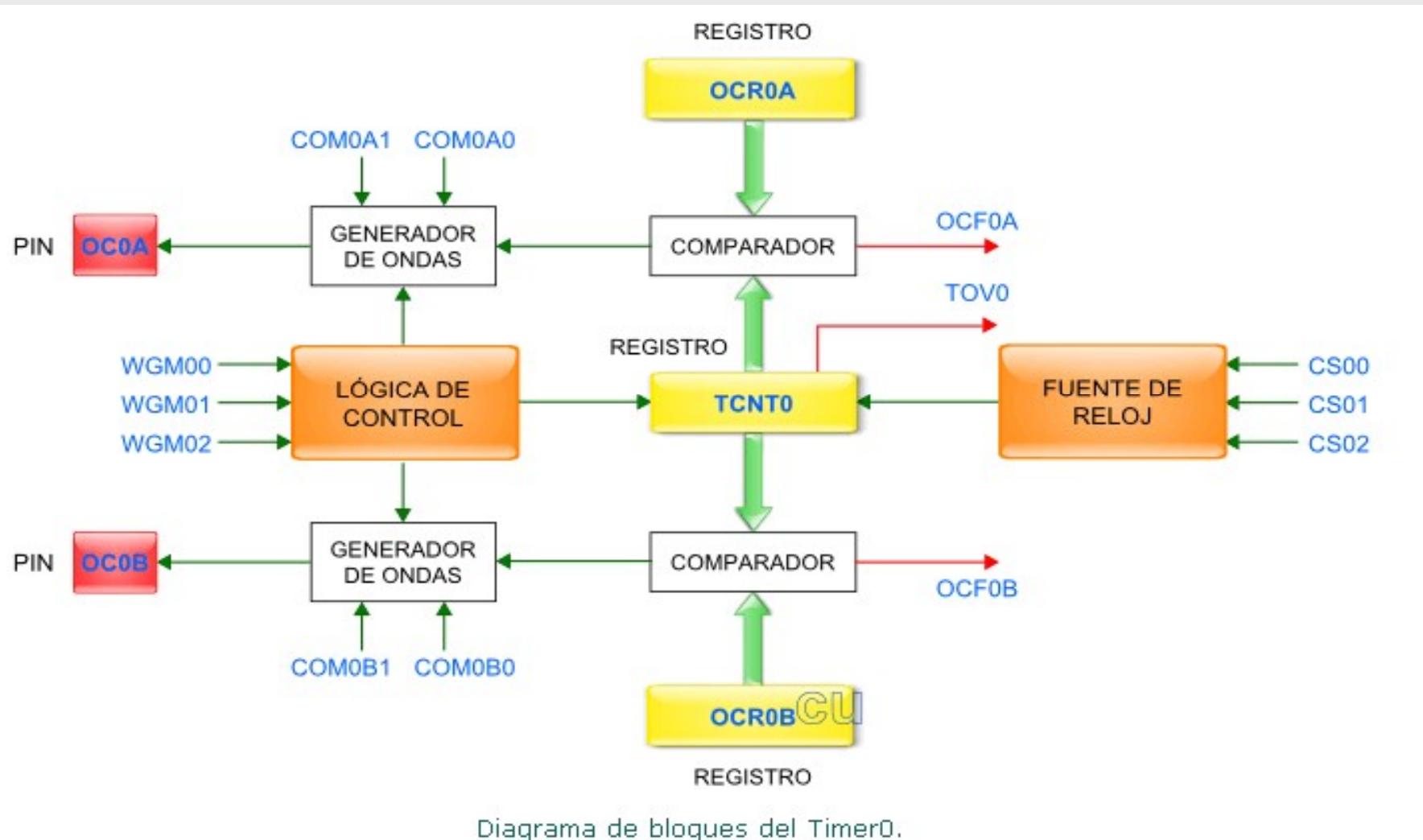
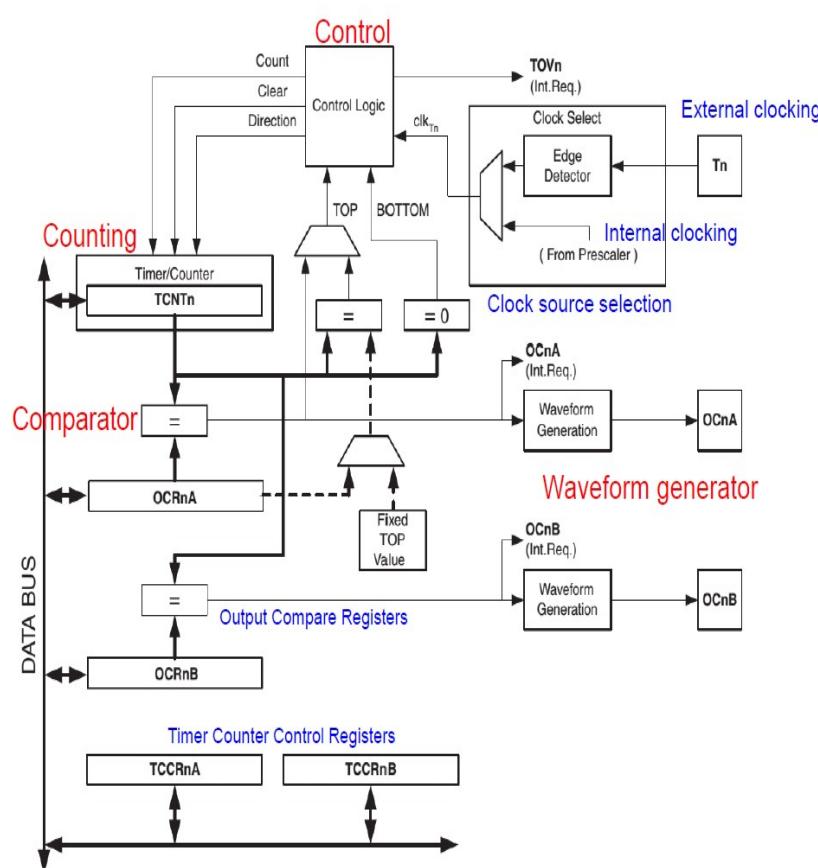
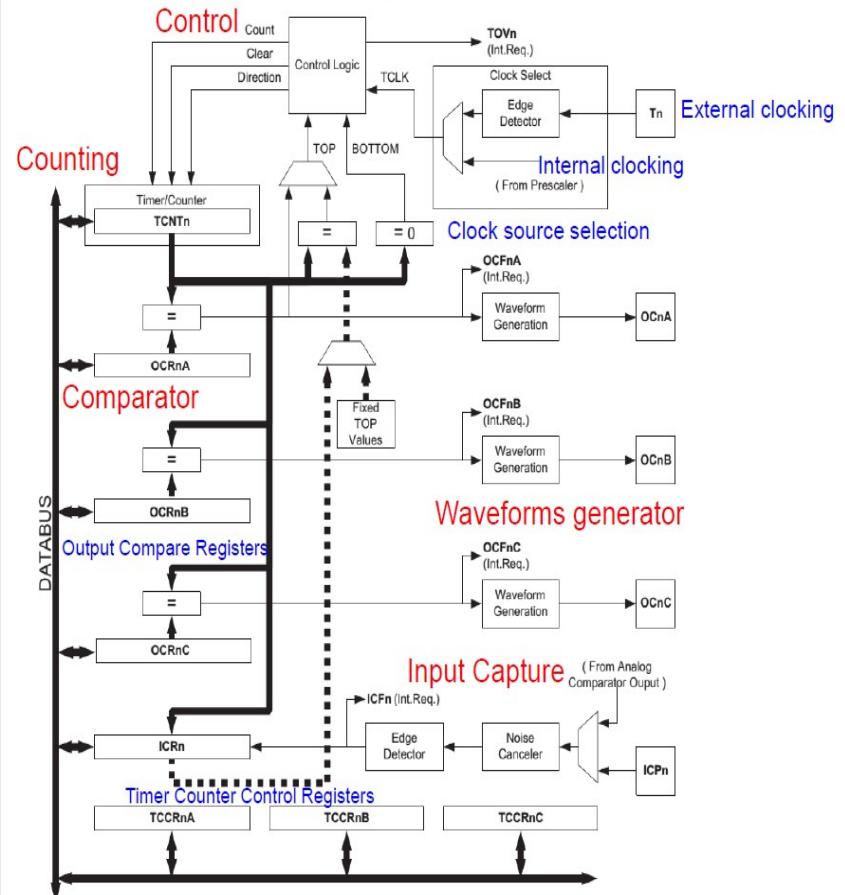


Diagrama de bloques timer 0 ATmega164P

Structure of 8 bit timers



Structure of 16 bit timers



REGISTROS DE CONTROL DEL TIMER / CONTADOR 0 ATmega164P

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	I/O	SRAM
TCCR0A	COM0A1	COM0AO	COM0B1	COM0BO	-	-	WGM01	WGM00	\$24	(\$44)
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	\$25	(\$45)

DESCRIPCIÓN	
FOC0A	Para obligar a una comparación A
FOC0B	Para obligar a una comparación B
WGM02/1/0	Para controlar la secuencia del Registro Contador
COM0A1/0	Modo de operación de la Salida A de Comparación
COM0B1/0	Modo de operación de la Salida B de Comparación
CS02/1/0	Bits de selección del reloj para el Contador

MODOS DE OPERACIÓN DEL TIMER / CONTADOR 0 ATmega164P

WGM02/1/0	DESCRIPCIÓN	FINAL	ENCENDIDO DE TOV0
000	Contador normal ascendente	0xFF	En el valor máximo
001	PWM de fase correcta	0xFF	Al inicio
010	Borrar el Timer al emparejar al comprador (CTC)	OCR0A	En el valor máximo
011	PWM rápido	0xFF	En el valor máximo
100	Reservado	---	---
101	PWM de fase correcta	OCR0A	Al inicio
110	Reservado	---	---
111	PWM rápido	OCR0A	En el valor máximo

COM01:0: Estos bits configuran el uso del pin OC0 con el modulo de comparación dependiendo del modo que se trabaje.

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

Salida del comparador en modo NO-PWM (Normal)

Salida del comparador en modo FAST PWM

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

Salida del comparador en modo NO-PWM (CTC)

Salida del comparador en modo PHASE CORRECT PWM

REG.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCCROA	COM0A1	COM0AO	COM0B1	COM0BO	-	-	WGM01	WGM00
TCCR0B	F0DA	F0DB	-	-	WGM02	CS02	CS01	CS00

Compare Output Mode (COM)

CTC or Normal
(Non PWM)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

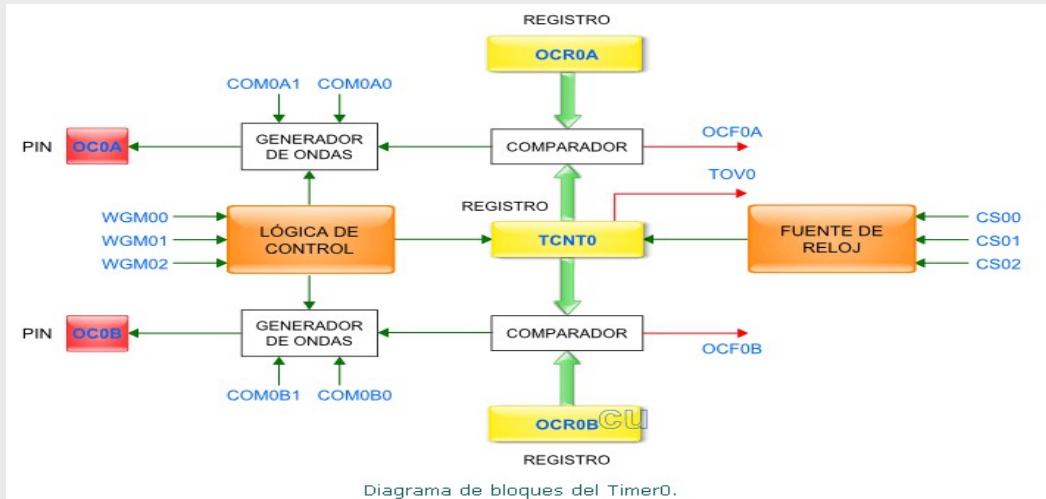
Fast PWM

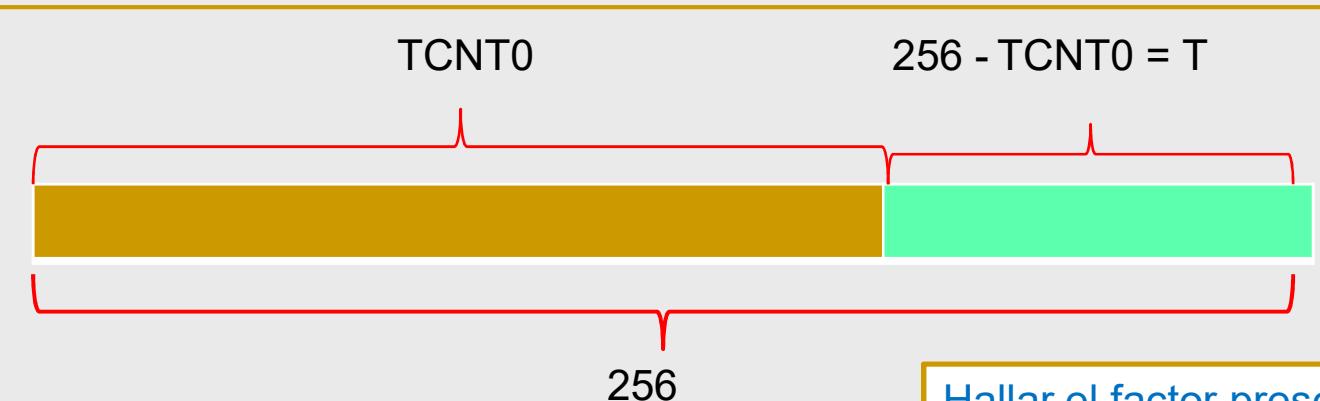
COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP
1	1	Set OC0 on compare match, clear OC0 at TOP

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 73 for more details.

Phase Correct
PWM

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.





Ejemplo

Tiempo = N*(256-TCNT0) / F_cpu

N = factor del Prescaler
(1,2,32,64,128...etc)

$$N = \text{tiempo} * F_{\text{cpu}} / 256$$

TCNT0= 256- (tiempo * Fcpu) / N

Hallar el factor prescaler N y el Valor TCNT0 para obtener 5 mseg con un Xtal de 10 MHz

$$N = 5 \text{ ms} * 10 \text{ MHz} / 256 = 195.31$$

→ 256

N 1,8,32,128,256

TCNT0=256- 5ms*10Mhz /256= 60.67 → 61

Timmer

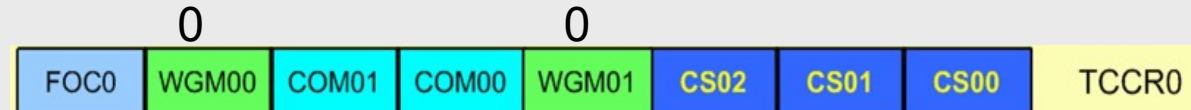
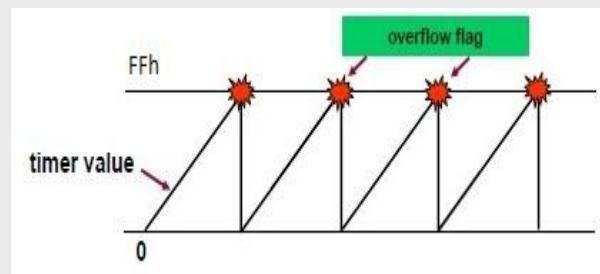
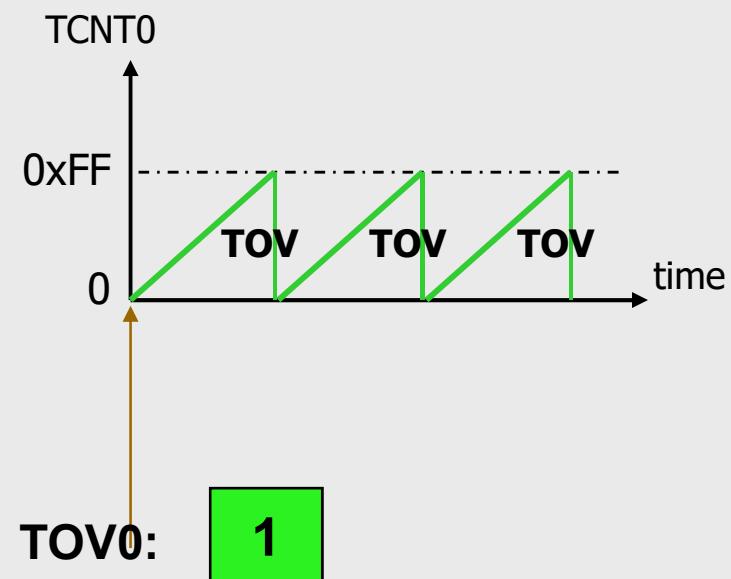
MODOS

- Modo Normal
- Modo Comparación / clear Timer (CTC)
- Modo PWM
- Modo rápido PWM

Modo Normal

1. Modo de conteo ascendente
2. Cuenta de 0x00 a 0xFF
3. Reinicia en 00 → T0V0 =1
4. T0V0 se comporta como 9º BIT , no se Borra por hard
5. Combinado con la interrupción de OV , TOVO se limpia con 1

El timer cuenta desde 0 a 255 y se desborda reiniciando la cuenta. Puede generar interrupción al desbordarse o cuando la comparación del conteo concuerde con un valor determinado



TOV0 = 1

Ejemplo 1: Escriba un programa que espera 14 ciclos de la máquina en el modo Normal.

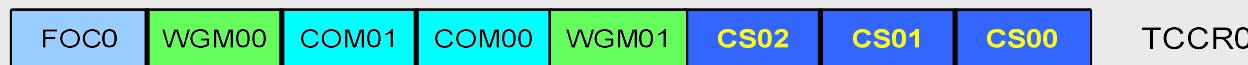
$14 = \$0E$



\$100

$-\$0E$

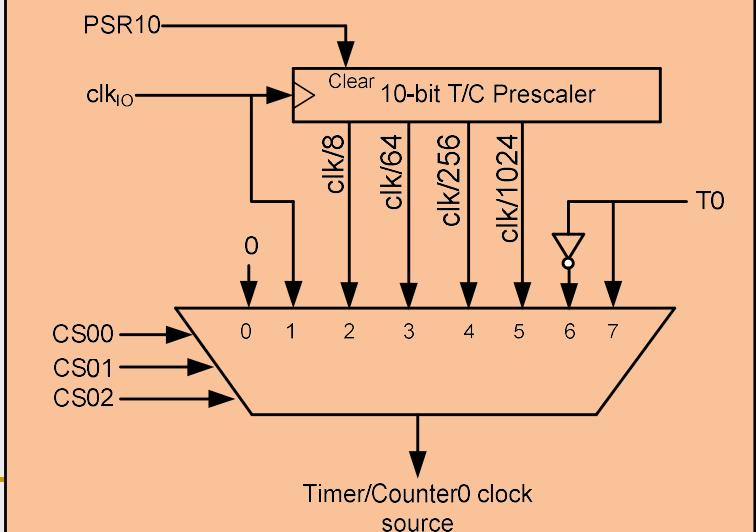
\hline
\$F2



WGM00	WGM01	Comment
0	0	Normal
0	1	CTC
1	0	PWM, phase correct
1	1	Fast PWM

\$100

1

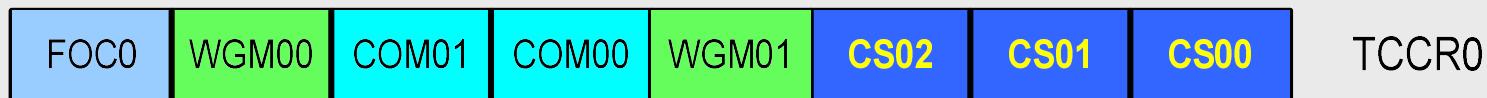


Programación del timer 0 en M. Normal

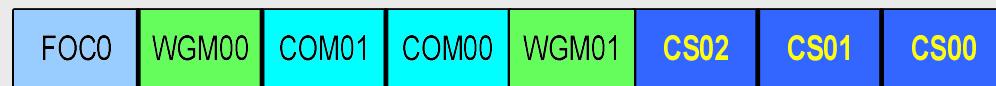
Timer0 AVR como temporizador

- Cargue el registro **TCNT0** con un valor inicial
- Configure el Registro **TCCR0**

```
BEGIN:    LDI    R20,0xF2
          OUT    TCNT0,R20 ;load timer0
          LDI    R20,0x01
          OUT    TCCR0,R20 ;Timer0,Normal mode ,int clk
```



```
.INCLUDE "M32DEF.INC"
```



TCCR0

```
LDI R16, 0x20
```

```
SBI DDRB, 5 ;PB5 as an output
```

Timer0 AVR como temporizador

```
LDI R17, 0
```

```
OUT PORTB, R17
```

```
BEGIN: LDI R20, 0xF2
```

```
OUT TCNT0, R20
```

;load timer0

```
LDI R20, 0x01
```

```
OUT TCCR0, R20
```

;Timer0,Normal mode,int clk

```
AGAIN: IN R20, TIFR
```

;read TIFR

```
SBRS R20, 0
```

;if TOV0 is set skip next inst.

```
RJMP AGAIN
```

```
LDI R20, 0x0
```

```
OUT TCCR0, R20
```

;stop Timer0

```
LDI R20, (1<<TOV0)
```

;R20 = 0x01

```
OUT TIFR, R20
```

;clear TOV0 flag

```
EOR R17, R16
```

;toggle D5 of R17

```
OUT PORTB, R17
```

;toggle PB5

```
RJMP BEGIN
```



TIFR

$$F_{Timer} = \frac{F_{CPU}}{\text{Prescaler}}$$

$$T_{Timer} = \frac{1}{F_{Timer}}$$

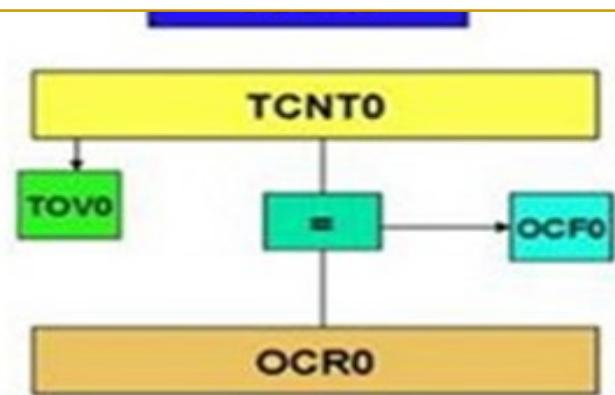
$$T_{overflow} = T_{Timer} \cdot \text{Resolución Timer}$$

- clock= 8 Mhz y un prescaler = 1
- , la frecuencia del timer sería:
- $F_t = 8000000 \text{ Hz} / 1$
- $T_{timer} = 1 / 8.000.000 \text{ Hz} = 0,125 \text{ microsegundos}$
- $T_{ovf} = 0,125 \text{ microsegundos} * 256 = 32 \text{ microseg}$
- 14 ciclos de maq
- $T_{ovf} = 0,125 \text{ microsegundos} * 14 = 1,75 \text{ microseg}$

- clock= 8 Mhz y un prescaler = 1024
- , la frecuencia del timer sería:
- $F_t = 8000000 \text{ Hz} / 1024 = 7812.5 \text{ Hz}$
- $T_t = 1 / 7812.5 \text{ Hz} = 128 \text{ microsegundo}$
- $T_{ovf} = 128 \text{ microsegundos} * 256 = 32.8 \text{ milseg}$
- 14 ciclos de maq
- $T_{ovf} = 128 \text{ microsegundos} * 14 = 1,79 \text{ milseg}$

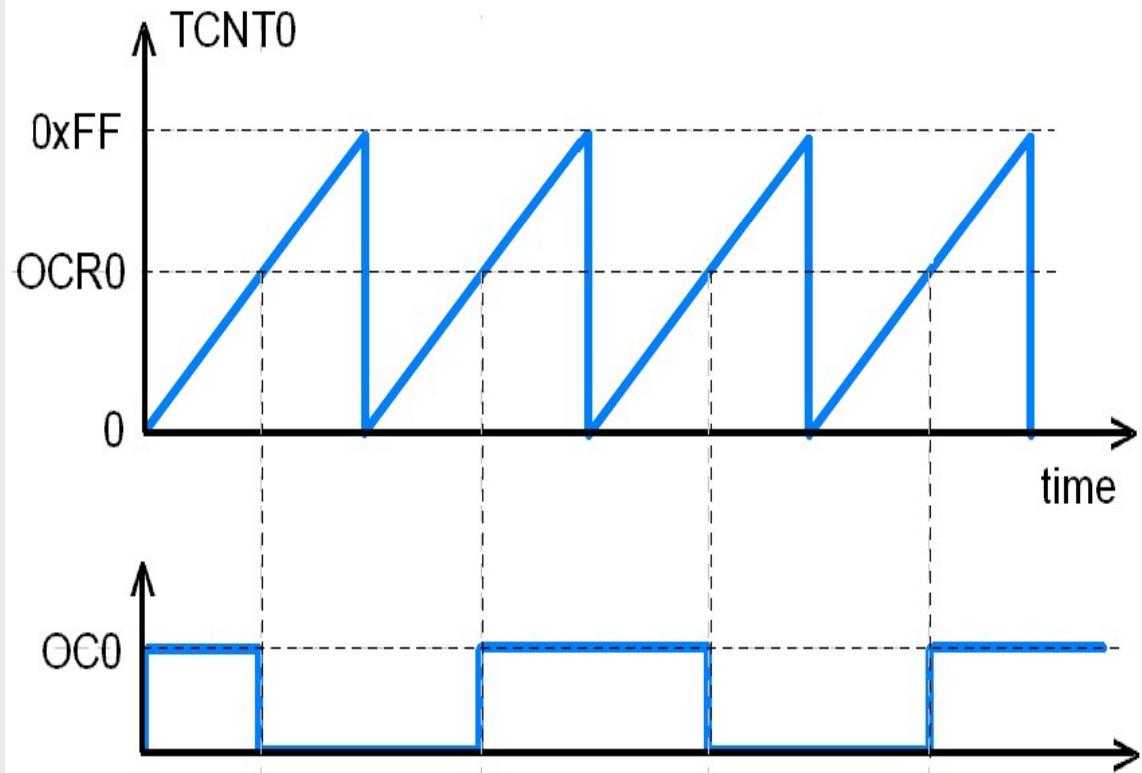
Waveform Generation Using Normal Mode

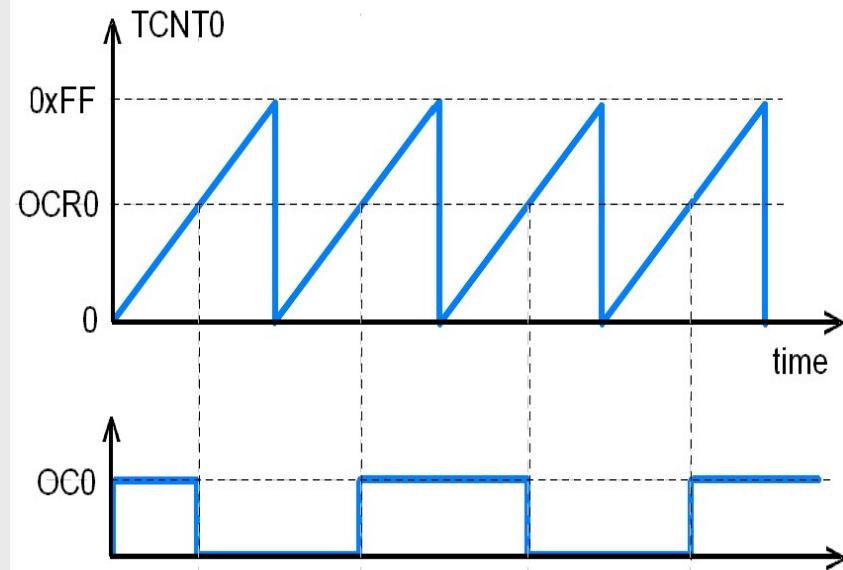
COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match



$$F_{Timer} = \frac{F_{CPU}}{\text{Prescaler}}$$

$$T_{Timer} = \frac{1}{F_{Timer}}$$





En el modo normal, cuando se produce una coincidencia, el pin OC0 alterna y el temporizador continúa contando hasta que alcanza el valor máximo.

Frecuencia de onda cuadrada:

Suponiendo una forma de onda $F_{osc} = 8MHz$,
 $T = 0.125 \mu s$.

Período de tiempo de la onda cuadrada: $2 \times 256 \times 0.125 \mu s = 64 \mu s$

Frecuencia de onda = $1/64 \mu s = 15,625 kHz$

Timer0 AVR como Gen. ondas

Assuming XTAL = 8 MHz, calculate the frequency of the wave generated by the following program:

```
.INCLUDE "M32DEF.INC"
    SBI    DDRB,3      ;PB3 as output
    LDI    R22,100
    OUT    OCR0,R22    ;set the match value
    LDI    R22,0x11    ;COM01:00 = Toggle, Mode = Normal, no prescaler
    OUT    TCCR0,R22   ;load TCCR0 and start counting
HERE: RJMP HERE
```

Solution:

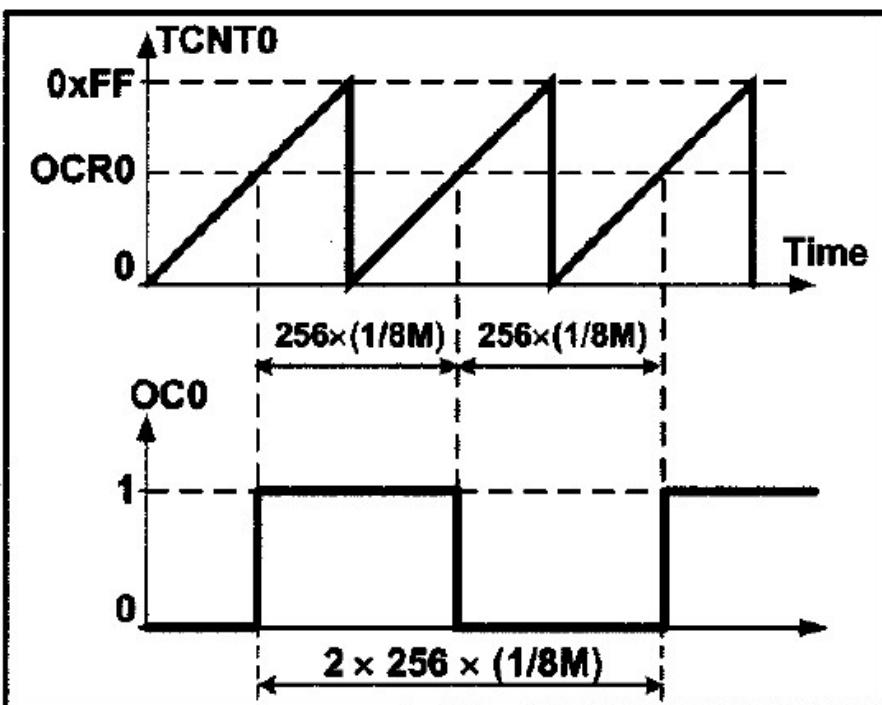
There are 256 clocks between two consecutive matches. Therefore

$$T_{\text{timer clock}} = 1/8 \text{ MHz} = 0.125 \mu\text{s}$$

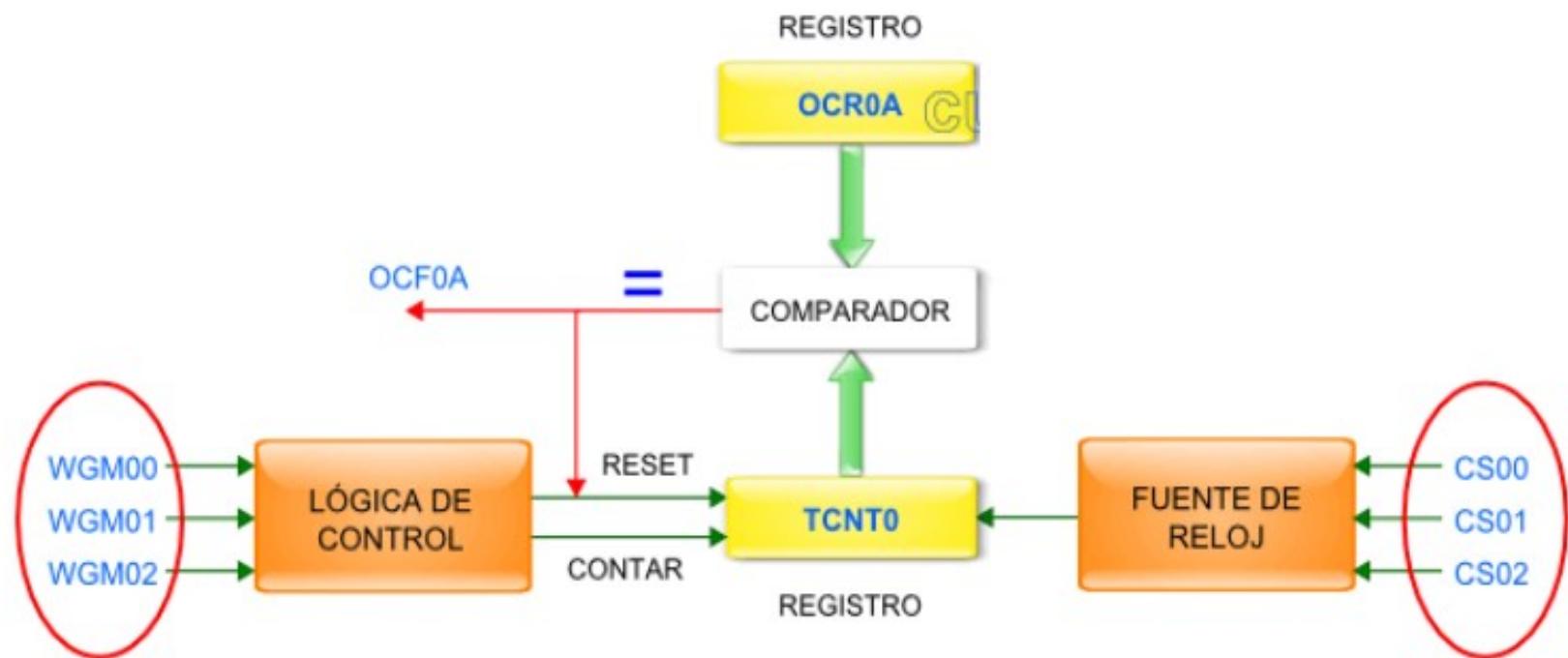
$$T_{\text{wave}} = 2 \times 256 \times 0.125 \mu\text{s} = 64 \mu\text{s}$$

$$F_{\text{wave}} = 1/64 \mu\text{s} = 15,625 \text{ Hz} = 15.625 \text{ kHz}$$

Note: In Normal mode, when match occurs, the OC0 pin toggles and the timer continues to count up until it reaches the top value.

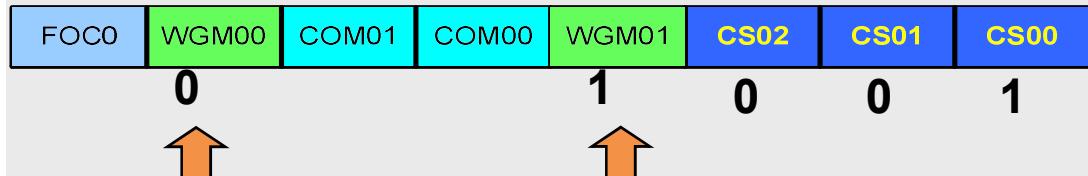


Modo CTC



FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

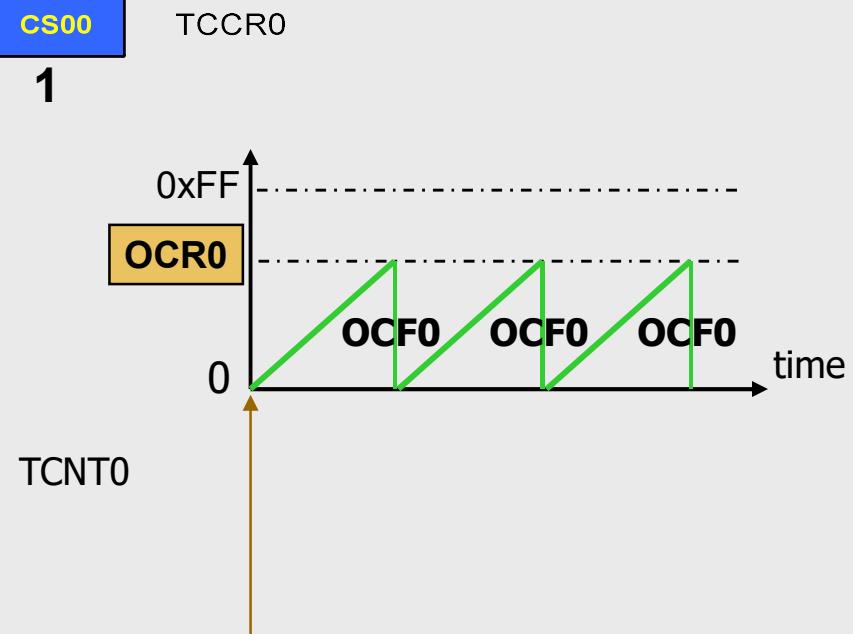
CTC (Clear Timer on Compare match)



En este modo el timer0 es reiniciado a 0 cuando una comparación entre el timer y un valor determinado coincide. Opcionalmente puede configurarse para que al haber una coincidencia genere una interrupción o cambie el estado de un pin

TOV0 = no change

OCF0 = 1



TOV0: 0

OCF0: 1

Square wave

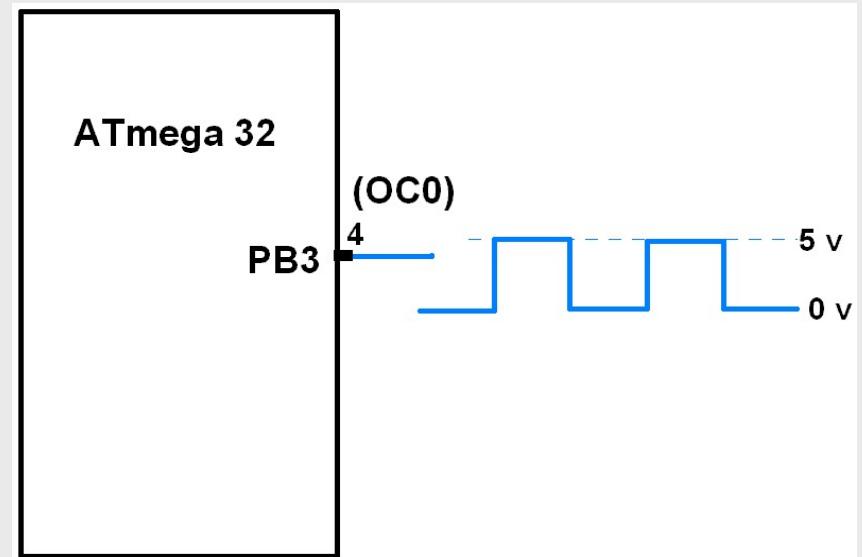
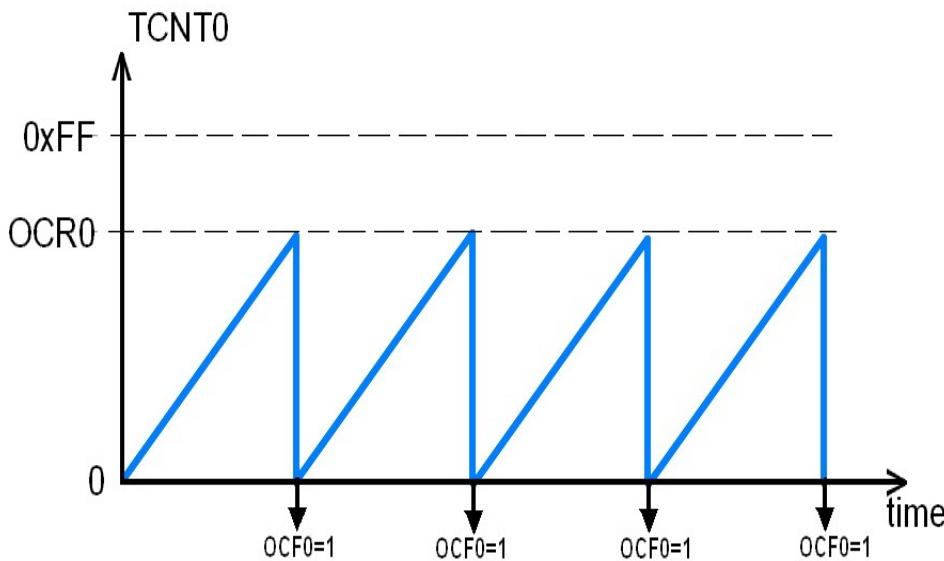
Using CTC mode:

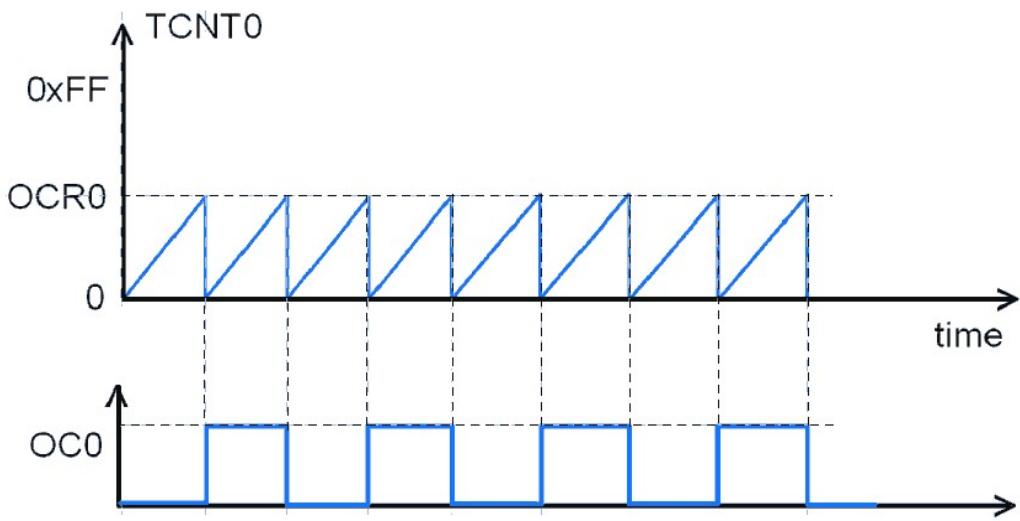
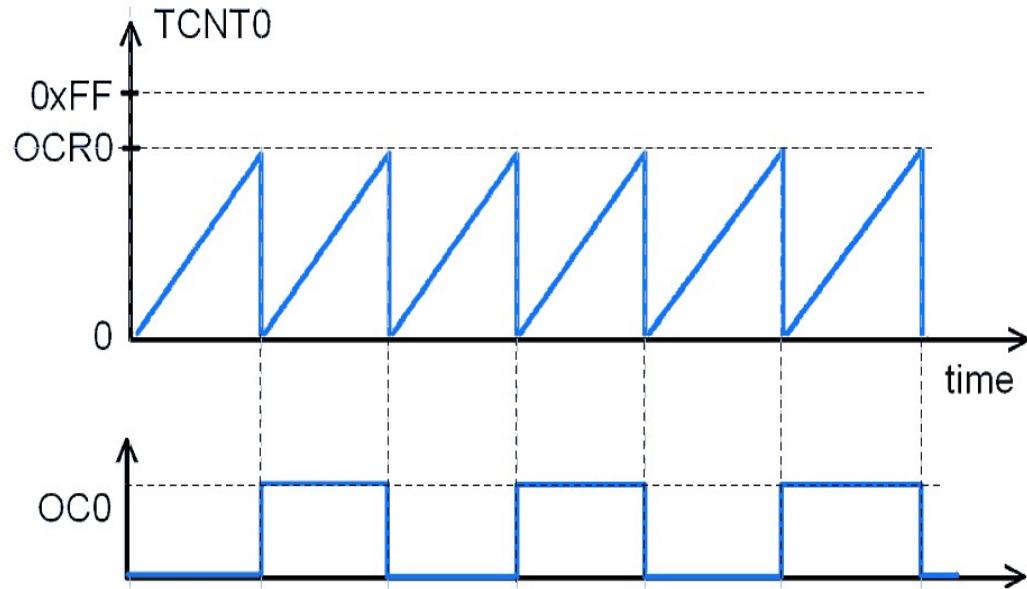
$$\text{Tiempo} = N * (\text{OCR0} + 1) / \text{F_clock}$$

$$N = (\text{Tiempo} * \text{F_clock}) / 256$$

$$\text{OCR0A} = (\text{Tiempo} * \text{F_clock}) / N - 1$$

WGM00	WGM01	Comment
0	0	Normal
0	1	CTC
1	0	PWM, phase correct
1	1	Fast PWM





Suponiendo una forma de onda cuadrada:
 $F_{osc} = 8\text{MHz}, T = 0.125 \mu\text{s}$.

Período de tiempo de onda cuadrada:

$$2 \times (\text{OCR0} + 1) \times 0.125 \mu\text{s}$$

$$= 2 \times 201 \times 0.125 \mu\text{s} = 50.25 \mu\text{s}$$

$$\text{Frecuencia de onda cuadrada} = \\ 1 / 50.25 \mu\text{s} = 19.9 \text{ KHz.}$$

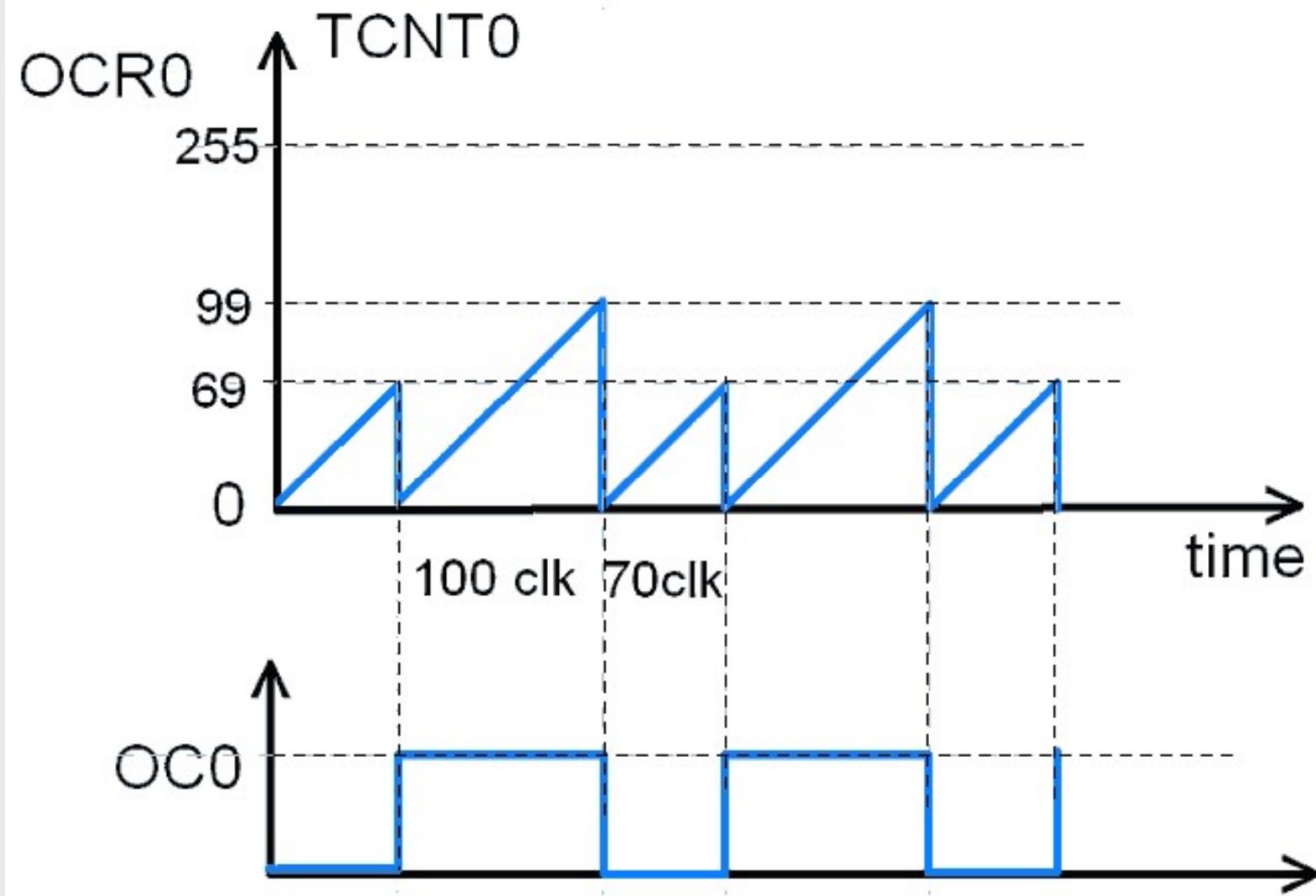
Período de tiempo de onda cuadrada:

$$2 \times (\text{OCR0} + 1) \times 0.125 \mu\text{s}$$

$$= 2 \times 100 \times 0.125 \mu\text{s} = 25 \mu\text{s}$$

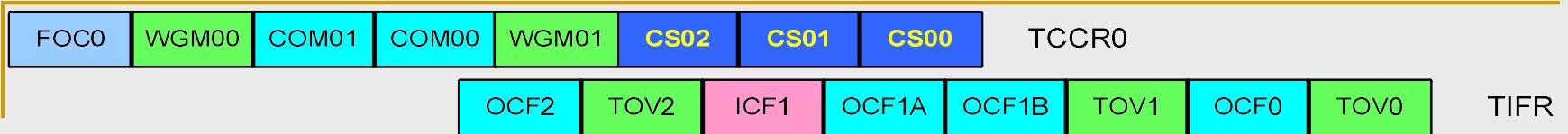
$$\text{Frecuencia de onda cuadrada} = \\ 1 / 25 \mu\text{s} = 40 \text{ KHz}$$

Modo CTC



Calculo para temporización en Modo CTC

- $Tiempo = N * (OCR0 + 1) / F_{clock}$
- $N = (Tiempo * F_{clock}) / 256$
- $OCR0A = (Tiempo * F_{clock}) / N - 1$



- Para una onda cuadrada con $T = 10$ microseg debemos tener un retardo de tiempo de 5 microseg. Debido $XTAL = 10$ MHz, el contador cuenta hacia arriba cada 0,1 ms. Esto significa que necesitamos $5\text{ ms} / 0,1\text{ microseg} = 50$ ciclos . Por lo tanto, tenemos $OCR0 = 49$.

```
.INCLUDE "M32DEF.INC"
```

```

LDI    R16,0x08
SBI    DDRB,3      ;PB3 as an output
LDI    R17,0
OUT   PORTB,R17
LDI    R20,49
OUT   OCR0,R20    ;load timer0
BEGIN: LDI   R20,0x09
        OUT  TCCR0,R20  ;Timer0,CTC
                           ;mode,intclk

```

Para una onda cuadrada con $T = 10$ ms debemos tener un retardo de tiempo de 5 ms. Debido a que XTAL = 10 MHz, el contador cuenta hacia arriba cada 0,1 ms. Esto significa que necesitamos $5\text{ ms} / 0,1\text{ ms} = 50$

AGAIN: IN	R20,TIFR	;read TIFR
SBRS	R20,OCF0	;if OCF0 is set skip next
RJMP	AGAIN	
LDI	R20,0x0	
OUT	TCCR0,R20	;stop Timer0
LDI	R20,0x02	
OUT	TIFR,R20	;clear TOV0 flag
EOR	R17,R16	;toggle D3 of R17
OUT	PORTB,R17	;toggle PB3
RJMP	BEGIN	

Examples

- **Example 1** – specified frequency waveform generation

```
.org 0x0000
jmp reset
```

reset:

```
ldi r16, 0b01000001
out TCCR0A, r16 ; configure Timer0A
```

```
ldi r16, 0b00000101
out TCCR0B, r16
```

```
ldi r16, 154 ; computed OCR
out OCR0A, r16
```

```
ldi r16, 0xff
out DDRx, r16 ; activates timer output OC0A
```

donothing:

```
rjmp donothing ; waveform can be monitored using an oscilloscope on OC0A
(PB7 for MEGA, PD6 for UNO) pin !!!
```

Mode	WGM00:01	Mode
0	00	Normal
1	10	PWM Phase Correct
2	01	CTC
3	11	Fast PWM

Normal, CTC

COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Toggle OC0 on compare match
10	Clear OC0 on compare match
11	Set OC0 on compare match

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stop)
0	0	1	clk _{I/O} /(No prescaling)
0	1	0	clk _{I/O} /8 (From prescaler)
0	1	1	clk _{I/O} /64 (From prescaler)
1	0	0	clk _{I/O} /256 (From prescaler)
1	0	1	clk _{I/O} /1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock
1	1	1	External clock source on T0 pin. Clock

Una señal PWM

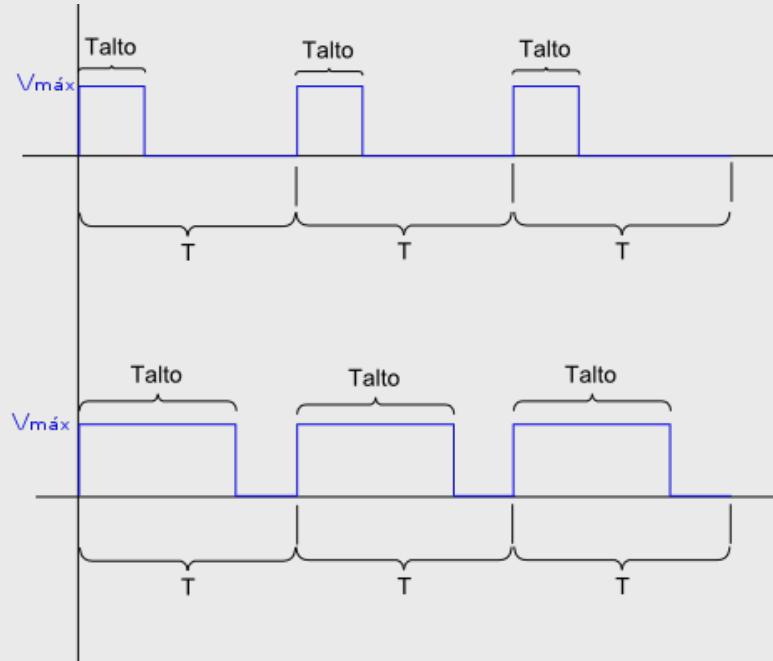
es una onda rectangular de periodo fijo y como la frecuencia es la inversa del periodo entonces también será de frecuencia fija, lo que si se cambiará o modificará normalmente en una señal PWM es el tiempo en alto que estará su valor máximo $V_{máx}$, a este tiempo en alto se le llama ancho de pulso, por lo que en una señal PWM se modificará su ancho pulso.

Si el ancho de pulso de la señal PWM se representa en forma de porcentaje se le suele llamar ciclo de trabajo, el ciclo de trabajo se obtienen mediante la siguiente relación:

$$\text{Ciclo de trabajo} = (\text{Talto}/T) * 100\%$$

El ciclo de trabajo puede ser desde un 0% cuando el Talto=0, hasta un 100% cuando el Talto es igual al periodo de la señal PWM $\text{Talto}=T$, si el tiempo en alto es igual a la mitad del periodo entonces el ciclo de trabajo será del 50%.

Al proceso de cambiar o modificar el tiempo en alto de la señal PWM es lo que se llama modificación por ancho de pulso, y por sus siglas del inglés se le llama PWM, ahora se verá como lograr señales PWM con el módulo PWM modo rápido timer0 AVR.



T es el periodo de la señal PWM el cual se mantiene constante, no se cambia.

Talto es el tiempo que la señal se mantiene en alto, este se puede variar o cambiar en una señal PWM.

Al tiempo que la señal se mantiene en alto se le conoce como ciclo de trabajo o también como ancho de pulso.

Como el ancho de pulso se puede variar de ahí viene el nombre de modulación por ancho de pulso, lo que es lo mismo que la modificación del ciclo de trabajo de la señal.

MODOS

■ Modo PWM

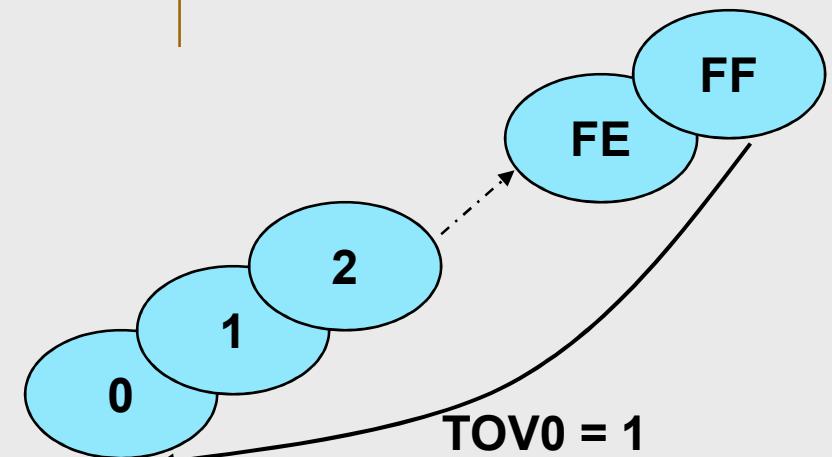
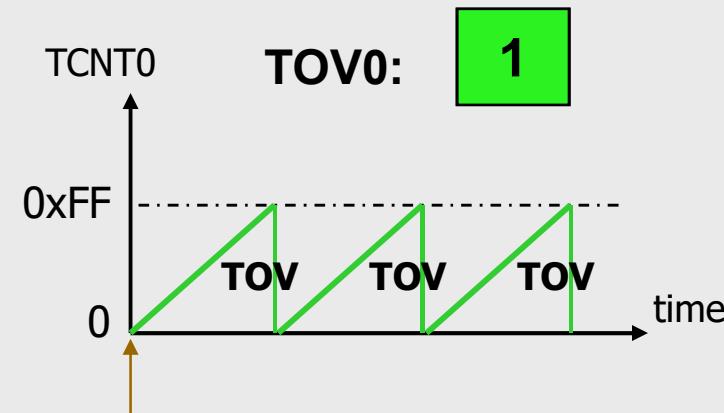
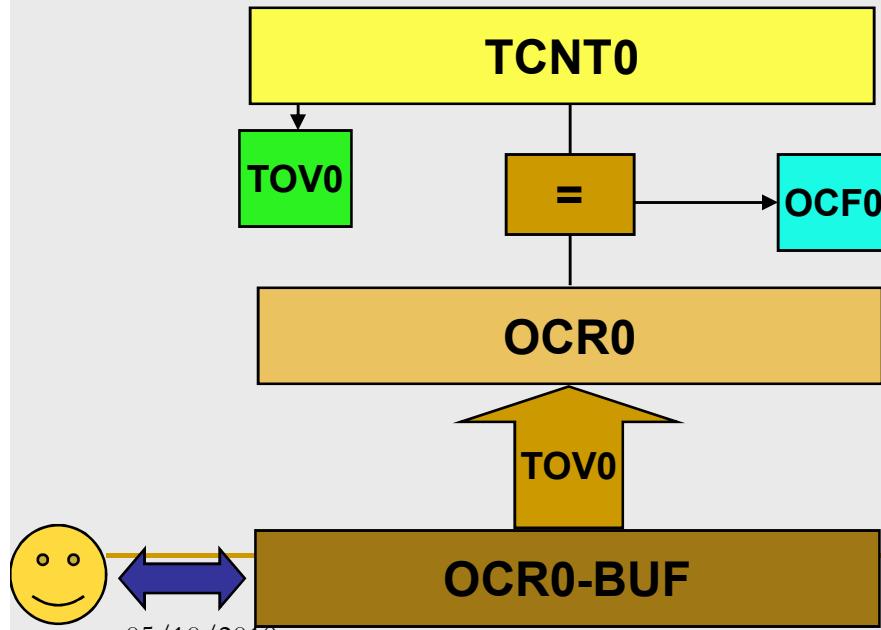
Este modo permite generar una onda PWM de alta frecuencia. El timer cuenta desde 0 a 255 y reinicia la cuenta. Con cada cuenta el valor del timer0 se compara con un valor determinado que cuando coinciden cambia el estado de uno de los pines de salida PWM, y cuando se reinicia el timer este pin vuelve a cambiar su estado.

■ Modo rápido PWM

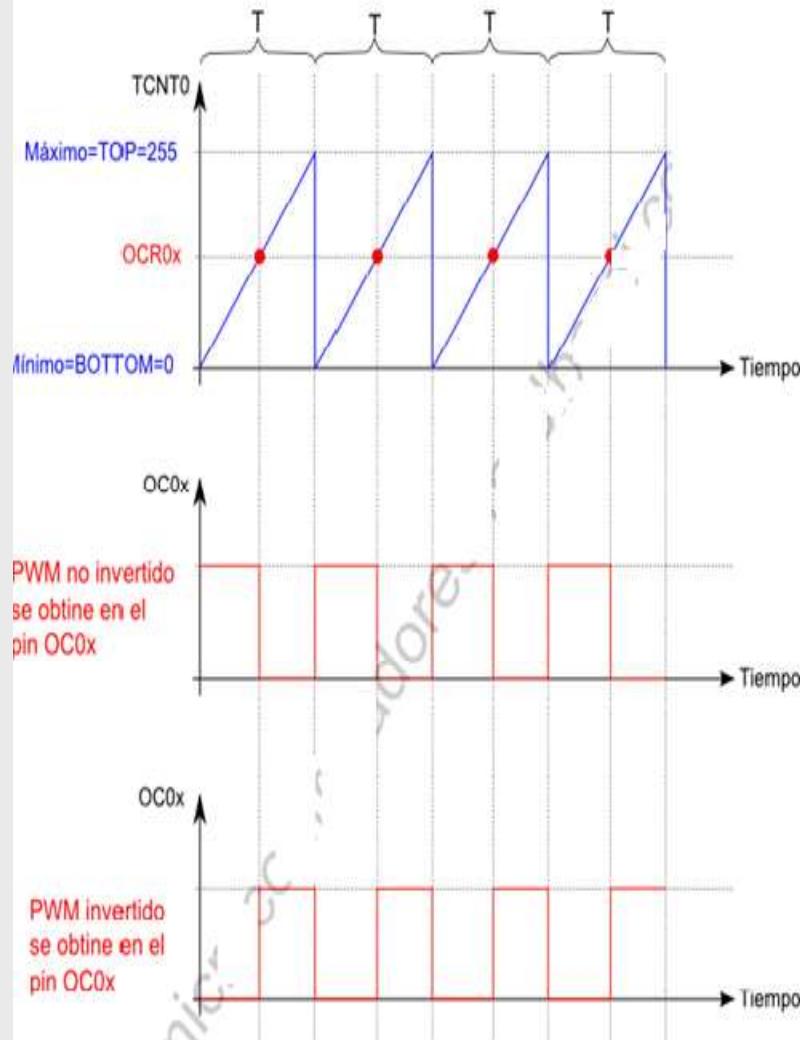
Este modo ofrece una onda PWM de alta resolución, a diferencia del modo Fast PWM, el timer cuenta hacia adelante y hacia atrás antes de hacer el cambio de estado del pin PWM, es decir cuenta de 0 a 255 al llegar a 255 cuenta de 255 a 0, obteniendo una salida PWM más limpia pero de menor frecuencia.

Fast PWM mode

- Similar al modo Normal OCR0 es buffered.



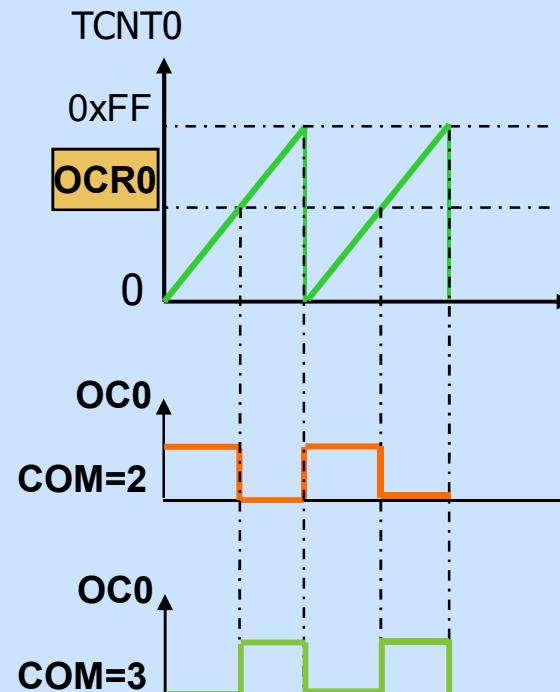
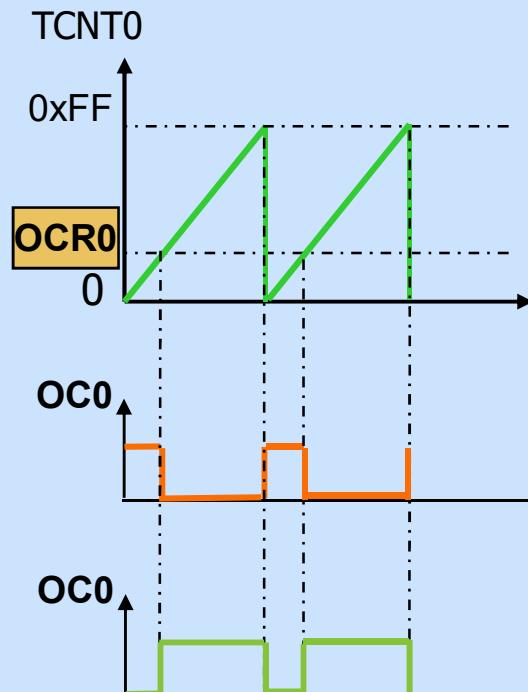
Cuando el valor del registro TCNT0 se iguale al valor del registro OCR0x, el estado de pin OC0x cambiará.



•**Para la forma no invertida** por el pin OC0x se obtendrá un alto mientras el valor del registro TCNT0 sea menor al valor almacenado en el registro OCR0x, al ocurrir la igualdad entre los registros TCNT0 y OCR0x el estado del pin OC0x cambiará a un bajo y se mantendrá así hasta que el registro TCNT0 llegue a su máximo que es 255 y vuelva 0, momento en el cual el estado del pin OC0x cambiará nuevamente a un alto y el ciclo se repetirá.

•**Para la forma invertida** por el pin OC0x se obtendrá un bajo mientras el valor del registro TCNT0 sea menor al valor almacenado en el registro OCR0x, al ocurrir la igualdad entre los registros TCNT0 y OCR0x el estado del pin OC0x cambiará a un alto y se mantendrá así hasta que el registro TCNT0 llegue a su máximo que es 255 y vuelva 0, momento en el cual el estado del pin OC0x cambiará nuevamente a un bajo y el ciclo se repetirá.

Fast PWM
Duty cycle = changeable (0% to 100%)
Frequency = selectable between limited choices

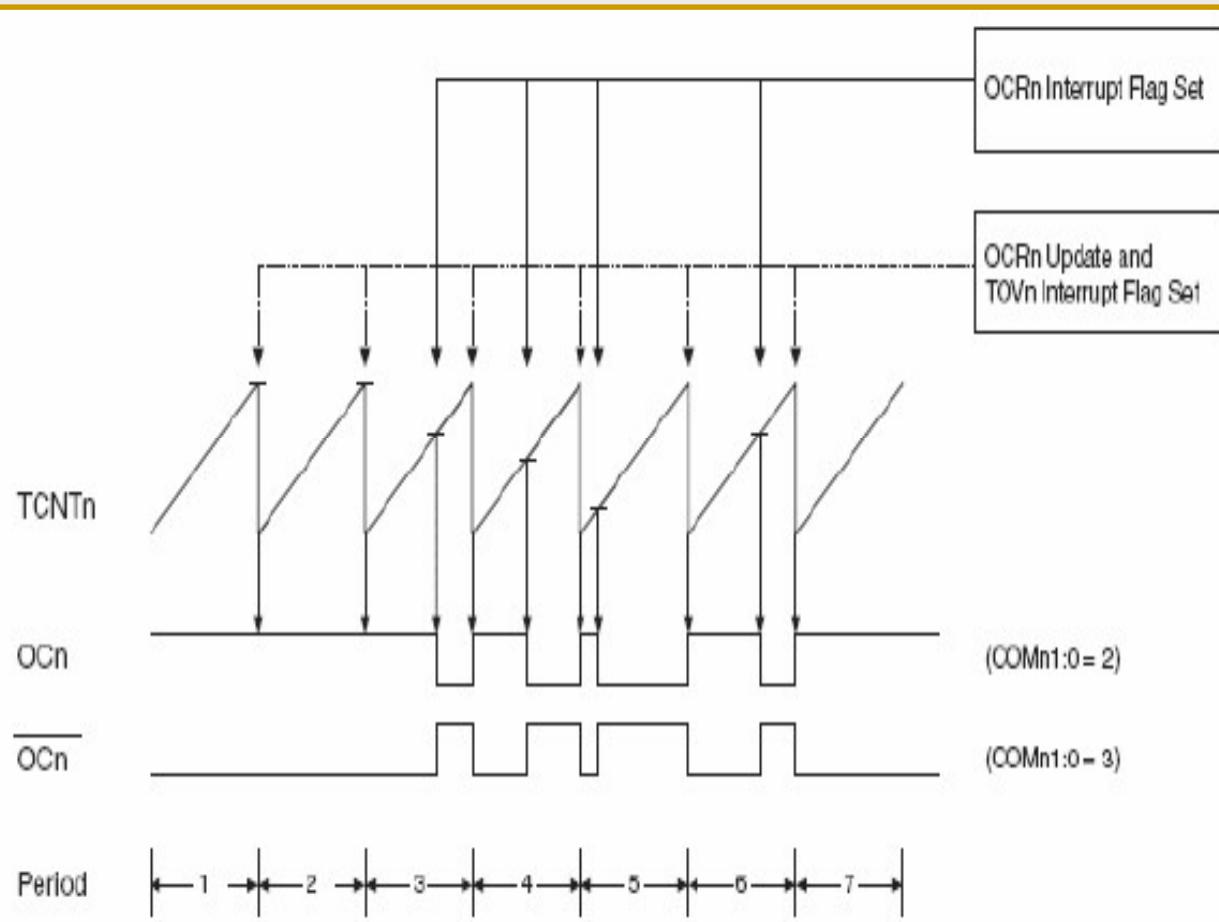


$$\text{Duty Cycle} = \frac{\text{OCR0} + 1}{256} \times 100$$

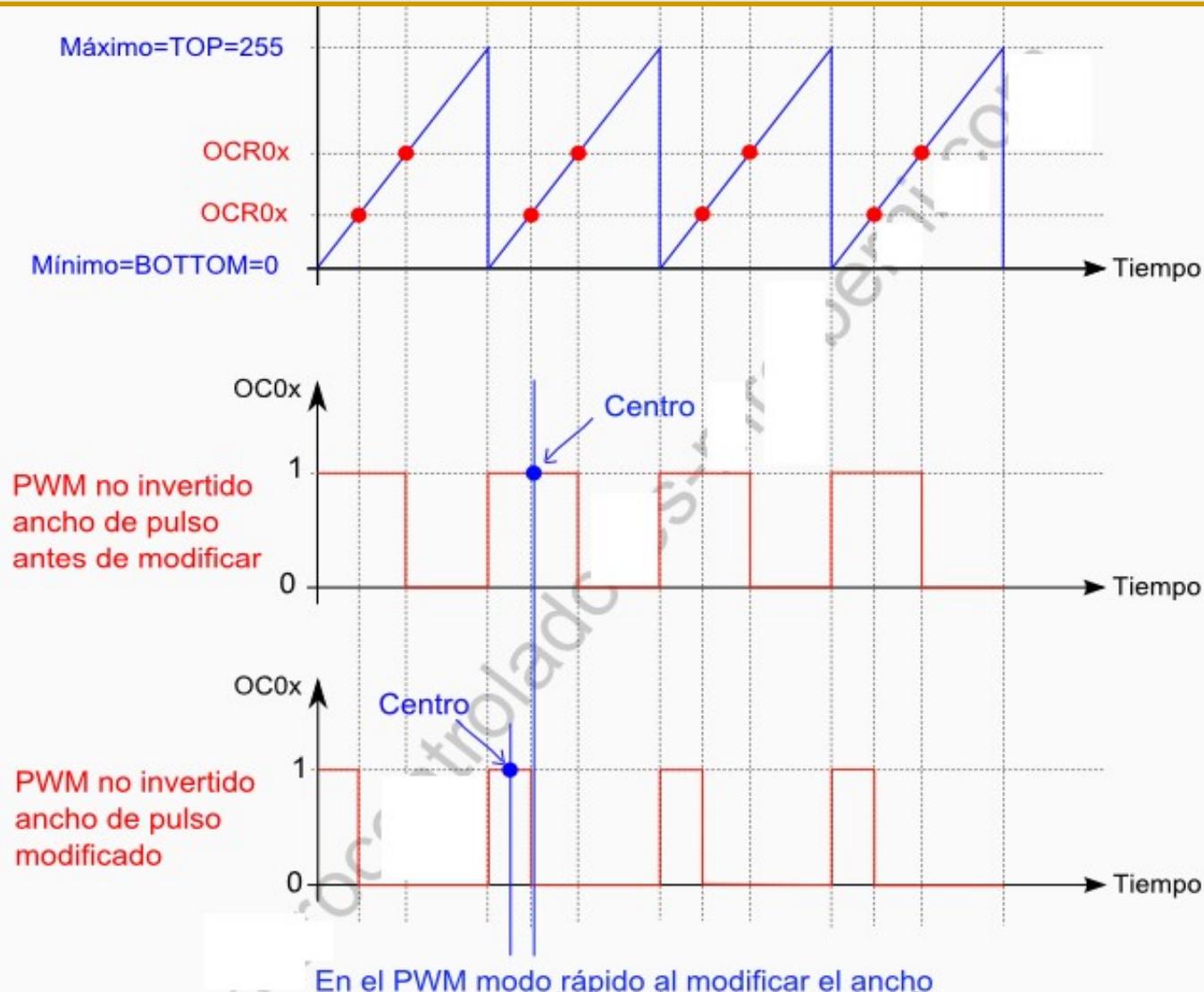
$$\text{Duty Cycle} = \frac{255 - \text{OCR0}}{256} \times 100$$

$$F_{Oco} = \frac{f_{clk}}{N(256)}$$

Fast PWM mode



Es importante señalar que el cambio en el valor almacenado en el registro OCR0x, el microcontrolador AVR lo hará cuando el registro TCNT0 llegue a su máximo valor no antes, esto es si TCNT0 se ha estado incrementando y en ese momento se modifica el valor a cargar en OCR0x, este no se modificará inmediatamente, sino que la modificación se realizará cuando TCNT0 llegue a su máximo.



Lo que diferencia al modo PWM rápido con el modo fase correcta, es que si se toma el centro de los anchos de pulso de la señal PWM timer0 AVR modo rápido y se hace un cambio en el ancho del pulso, el centro del ancho de pulso se moverá, cambiará de lugar, se dice que se modifica la fase del ancho de pulso, o que la fase no es correcta.

XTAL = 8 MHz, duty cycle = 75% y frecuencia = 31.250KHz
configurar

$$\text{Duty Cycle} = \frac{\text{OCR0} + 1}{256} \times 100$$

$$F_{\text{OC0}} = \frac{f_{\text{clk}}}{N(256)}$$

$$F_{\text{OC0}} = \frac{f_{\text{clk}}}{N(256)} \rightarrow 31.250\text{KHz} = \frac{8\text{MHz}}{N(256)} \rightarrow N = \frac{8\text{MHz}}{31.250\text{K} \times 256} = 1$$

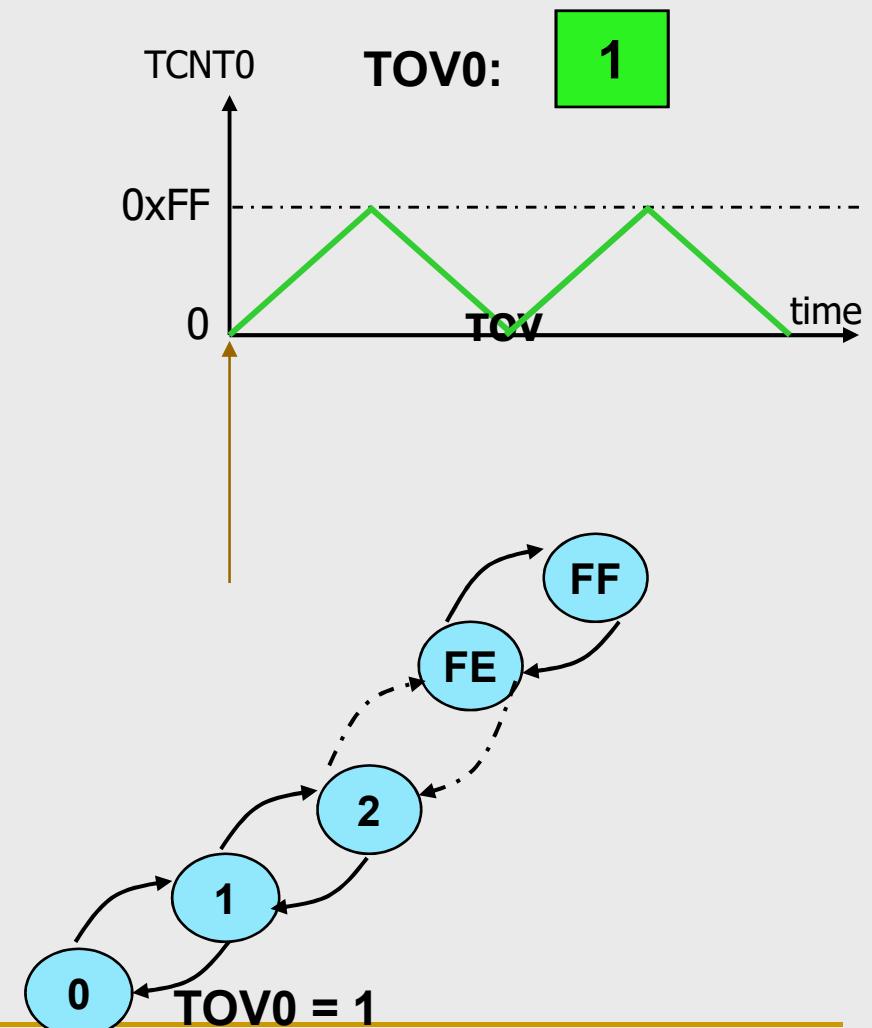
$$75/100 = (\text{OCR0}+1)/255 \rightarrow \text{OCR0}+1 = 191 = 0xBF \rightarrow \text{OCR0} = 0xBE$$

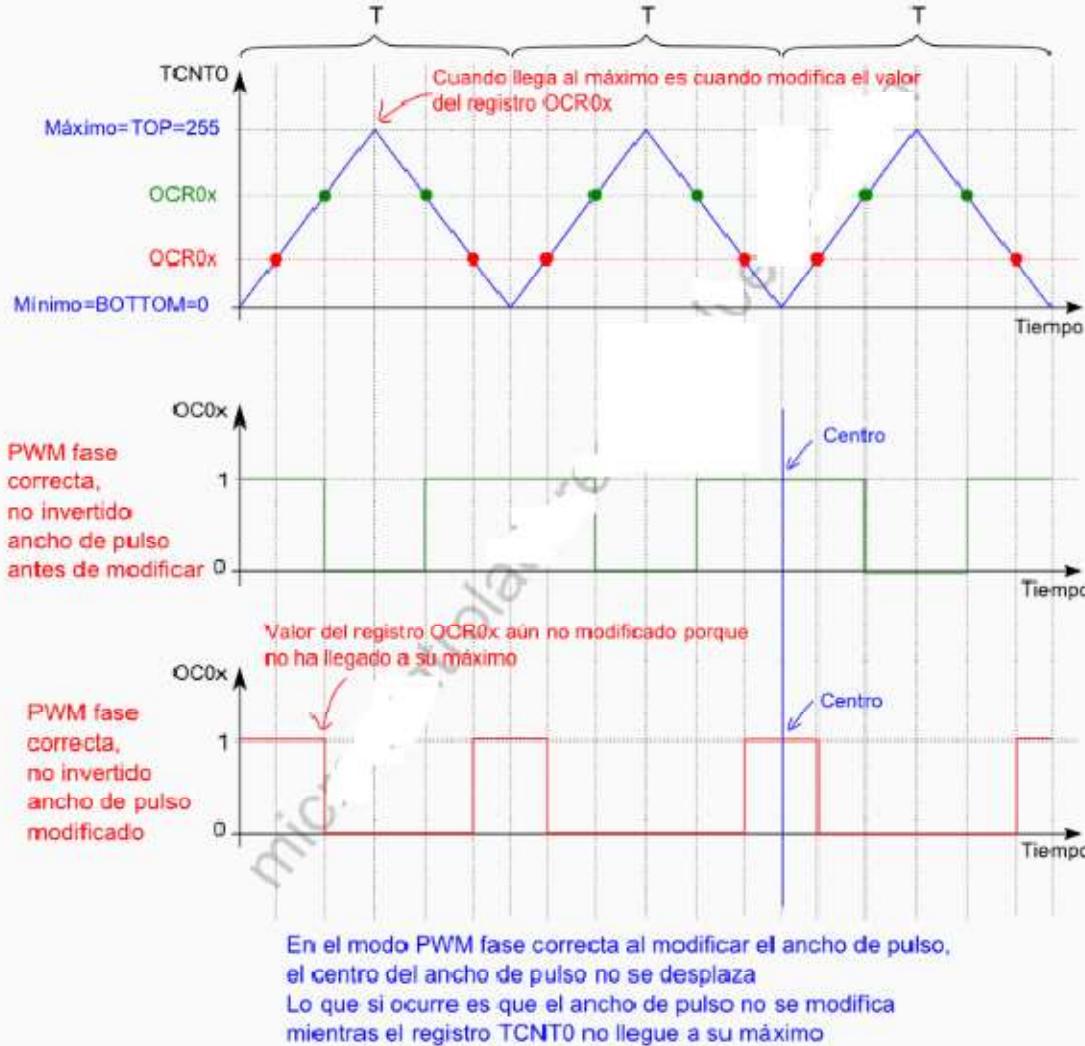
```
LDI R20, 0xBE  
OUT OCR0, R20  
LDI R20, 0x79  
OUT TCCR0, R20
```

```
OCR0 = 0xBE;  
TCCR0 = 0x79;
```

Phase Correct PWM mode

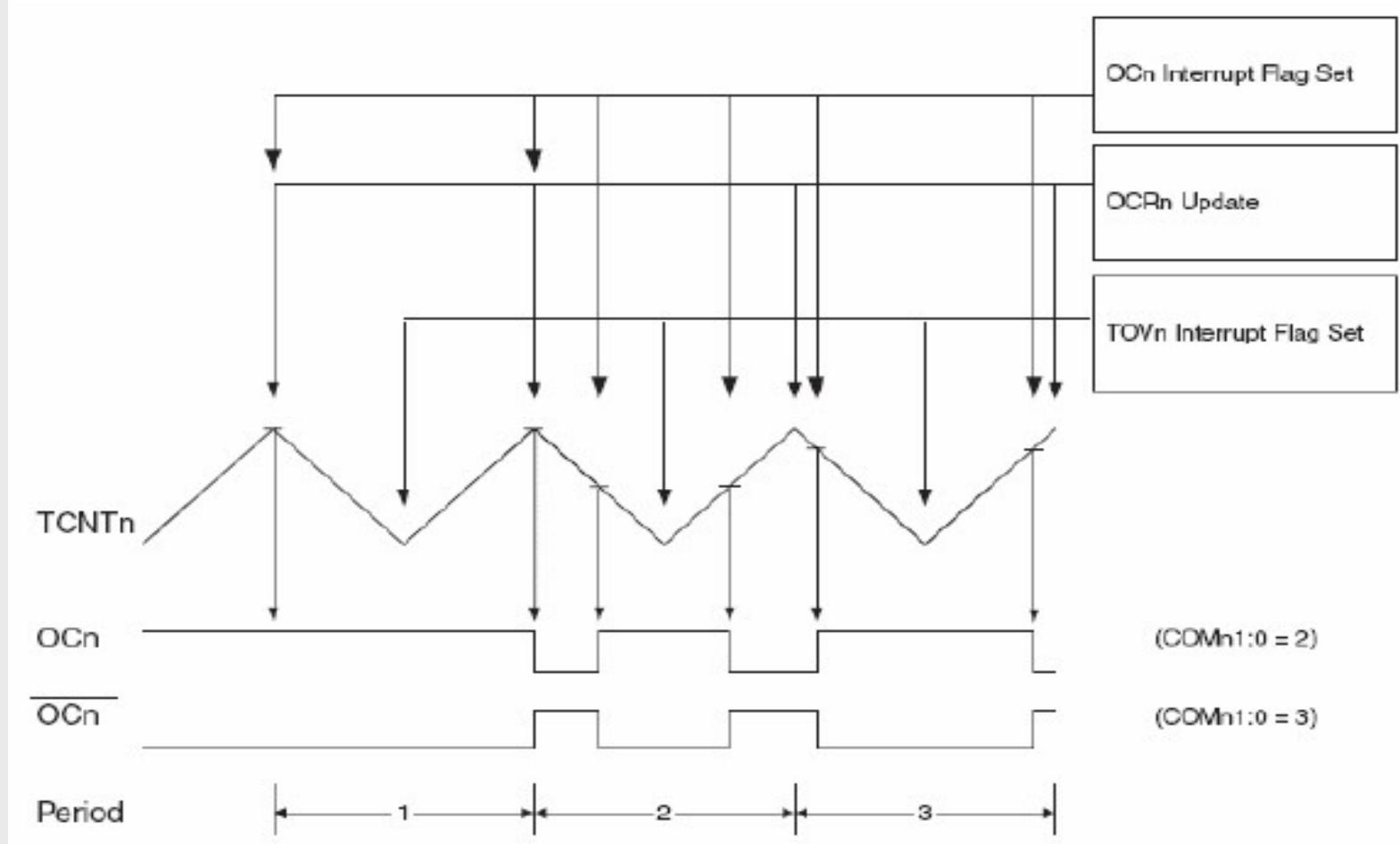
- En el PWM timer0 AVR fase correcta el registro TCNT0 irá de 0 a 255 y luego de 255 a 0, luego otra vez de 0 a 255 y nuevamente de 255 a 0, y así continuará en el modo fase correcta; a este ir de 0 a 255 y de 255 a 0 le tomará un tiempo que dependerá del prescaler utilizado para el timer0, ese tiempo que le tome al registro TCNT0 para ir de 0 a 255 y de 255 a 0 es el periodo de la señal PWM timer0 AVR fase correcta, al registrar triangular en el transcurso del tiempo.
- Una cosa a tener en cuenta es que al pasar el registro TCNT0 de 0 a 255 y luego de 255 a 0 en este caso se dispara la bandera de interrupción por desborde cuando el temporizador va de 255 a 0, por lo que se puede habilitar el uso de la interrupción por desborde del timer0.





Una observación a tener muy en cuenta y es lo que diferencia al modo PWM rápido con el modo fase correcta, es que si se toma el centro de los anchos de pulso de la señal PWM timer0 AVR fase correcta y se hace un cambio en el ancho del pulso, el centro del ancho de pulso no se moverá, no cambiará de lugar, se dice que la fase del ancho de pulso no se modifica, o que la fase es correcta

Phase Correct PWM mode



EL ciclo de trabajo de la señal PWM timer0 AVR fase correcta.

Ciclo de trabajo=((2*OCR0x)/510)*100%

Ciclo de trabajo=((OCR0x)/255)*100%

Por ejemplo si se quiere un ciclo de trabajo del 15%, se puede proceder así

$$15\% = ((OCR0x)/255) \times 100\%$$

De donde al despejar OCR0x se tendrá:

$$OCR0x = 38$$

Esto indica que si se quiere un ciclo de trabajo del 15% de la señal PWM timer0 AVR fase correcta en el registro OCR0x se tendrá que cargar el valor de 38 y esto será lo que se haga en el primer ejemplo.

Si se utilizan ambos pines OC0A y OC0B, la frecuencia de las señales PWM obtenidas en ambos pines serán las mismas, ya que ambas utilizan el mismo registro TCNT0, que es con el que decide la frecuencia o su inversa que es periodo de la señal PWM.

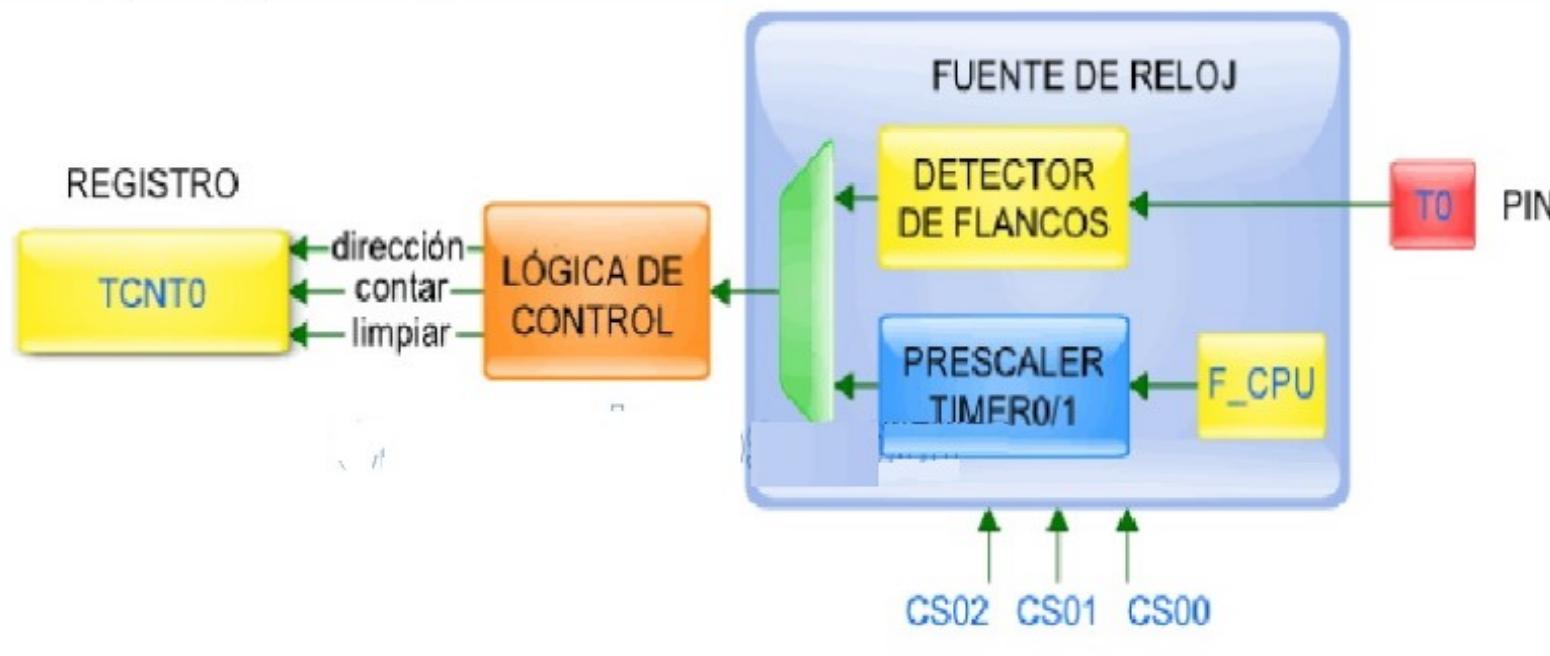
Timer como contador externo

El modo donde el Timer0/2 trabaja con un reloj externo aplicado al pin T0 (para el Timer0) o TOSC1 (para el Timer2) se conoce como modo Contador porque de alguna forma el Timer contará los pulsos detectados en dicho pin.

Sin embargo, el hecho de que el reloj provenga de una fuente externa no le quita sus otras funcionalidades, como por ejemplo, poder generar ondas PWM, interrupciones, etc., claro que sería conveniente que para tal caso la señal fuera periódica.

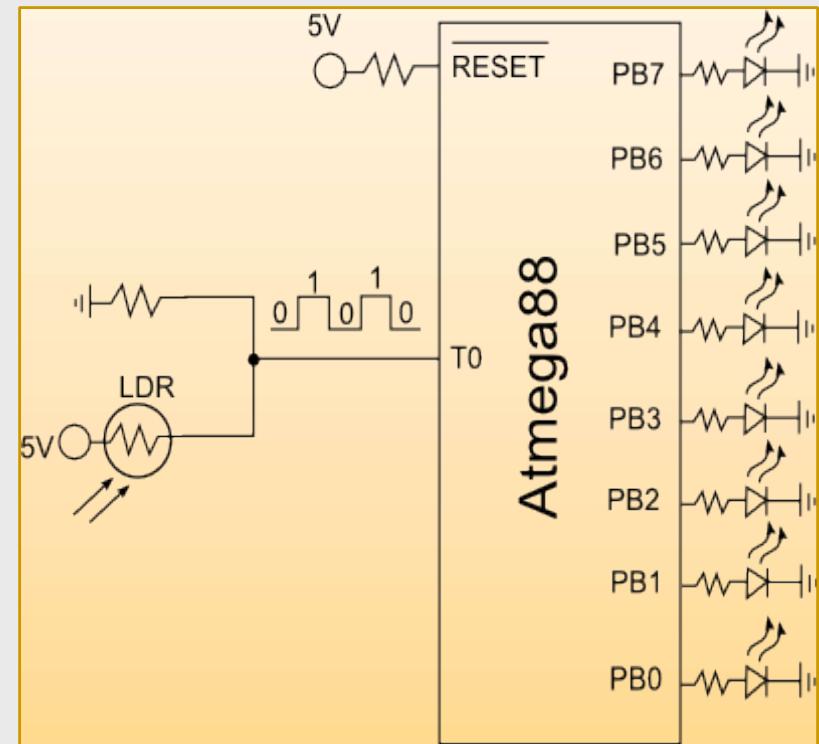
Timer como contador externo T0

CS02	CS01	CS00	Fuente de reloj del Timer0
1	1	0	Reloj externo en pin T0. El Timer0 avanza en el flanco de bajada.
1	1	1	Reloj externo en pin T0. El Timer0 avanza en el flanco de subida.



Ejemplo

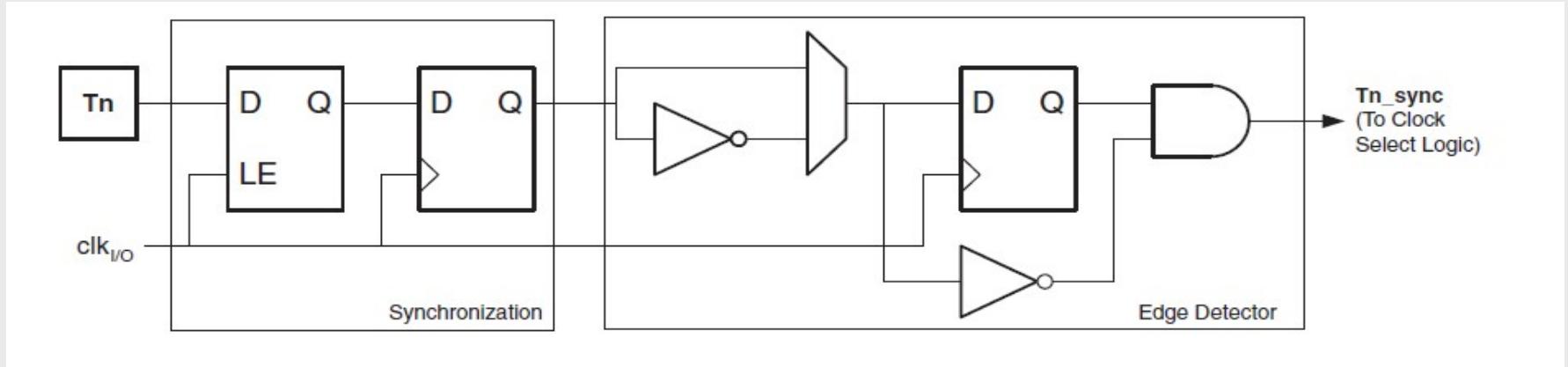
Se utilizará el timer0 AVR como contador, para ello se ha preparado el circuito tal como se muestra en la imagen, la señal digital será generada con la ayuda de un fotorresistor y una resistencia conectados al pin T0, se hará que le registro TCNT0 aumente su cuenta por cada flanco de subida, cada vez que al se ilumine el fotorresistor este disminuá su valor y al pin T0 le llegará un alto produciendo un flanco de subida, lo que provocará que el registro **TCNT0** aumente en una unidad, cuando se deje de iluminar el fotorresistor este aumentará su valor por lo que al pin T0 le volverá a llegar un 0, pero esto no provocará el aumento del registro TCNT0 en una unidad, ya que será un flanco de bajada; los valores del registro TCNT0 serán vistos en el puerto B por medio leds, estos valores serán vistos por tanto en forma binario contando de 0 a 255, tras lo cual se reiniciará para volver a contar de 0 a 255.



Timer como contador externo T0

```
DDRB=0xff;//se utiliza el puerto b para ver los valores del timer0  
PORTB=0;//el puerto b se pone a cero  
TCCR0A=0;//este registro a cero  
TCNT0=0;//el contador a cero  
TCCR0B |=(7<<1);//CS00=1, CS01=1 y CS02=1 conteo por flanco de subida
```

TEMPORIZACIÓN EXTERNA T0/1



- Se puede utilizar una fuente de reloj externa aplicada al pin T1 / T0 como clock temporizador / contador
- El pin T1 / T0 se muestrea una vez en cada ciclo de reloj del sistema mediante la sincronización de pin
 - La señal sincronizada (muestreada) pasa luego a través del detector de flancos
 - Los registros están sincronizados en el flanco positivo del reloj interno del sistema
 - El detector de flancos genera un pulso clkT0 para cada flanco positivo o negativo que detecta.
 - La sincronización y la lógica del detector de bordes introducen un retraso de 2.5 a 3.5 ciclos de reloj del sistema
 - No se puede prescalar

TIMER-2

No funciona como contador,

Funciona en :

- modo temporizador,
- modo comparador,
- modo PWM,

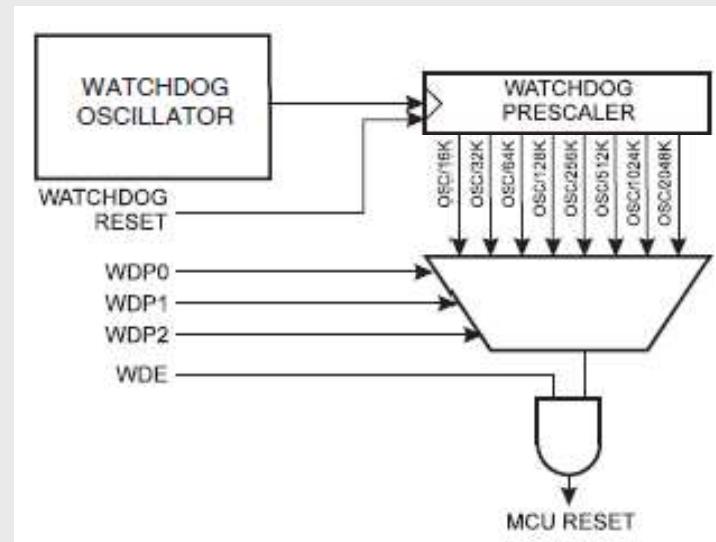
TIMER-2 como temporizador,

- Trabaja con el mismo clock interno
- Clock independiente conectado exteriormente a los pines TOSC1 y TOSC2, en este caso se dice que trabaja en forma asíncrona al sistema.
- El hardware se ha optimizado para una frecuencia de 32.768 KHz → múltiplos de segundos

WATCHDOG TIMER.

(PERRO GUARDIÁN)

Este periférico permite que el microcontrolador se reinicie automáticamente mediante un circuito contador, que se incrementa continuamente a una frecuencia de 1MHz aproximadamente. Para evitar que el microcontrolador se reinicie, hay que restablecer el valor del contador antes de que este se desborde. El restablecimiento del contador se efectúa mediante una instrucción especial (WDR).



NOMBRE	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0

WDP2	WDP1	WDP0	INTERVALO DE TIEMPO A 5 v (mili segundos)	INTERVALO DE TIEMPO A 3 v (mili segundos)	NÚMERO DE CICLOS DEL CONTADOR INTERO DEL PERRO GUARDIAN. (K CICLOS)
0	0	0	15	47	16
0	0	1	30	94	32
0	1	0	60	190	64
0	1	1	120	380	128
1	0	0	240	750	256
1	0	1	490	1500	512
1	1	0	970	3000	1024
1	1	1	1900	6000	2048

Tabla WDT-2.

Secuencia de configuración

1 Deshabilite Interrupciones Globales. (Recomendado pero no necesario)

SREG<I>="0"

2 Seleccione el intervalo de tiempo. Ver Tabla WDT-2.

3 Habilite el Perro Guardián.

WDTCR<WDE>="1"

A partir del momento de habilitar al Perro Guardián, cuenta con el intervalo de tiempo especificado en la Tabla WDT-2 , para ejecutar una instrucción WDR. En caso de NO ejecutar la instrucción WDR, el microcontrolador se reiniciará.

4 Habilite interrupciones globales.

SREG<I>="1"

;PASO1

cli ;SREG<I>=0. int. desabilitadas.

;PASO2

ldi r17, 0

out WDTCR, r17 ;Watch Dog Timer se
;reinicia cada 15 ms.

;PASO3

in r17, WDTCR

ori r17, (1<<WDE) ;Habilita Watch Dog Timer

out WDTCR, r17

;PASO4

sei

loop:

nop

nop

nop

rjmp loop

WDR

Que pasa??????

WDR – Watchdog Reset

(i) **Operation:**
WD timer restart.

(i) **Syntax:** WDR **Operands:** None **Program Counter:** PC \leftarrow PC + 1

16-bit Opcode:

1001	0101	1010	1000
------	------	------	------

Status Register and Boolean Formula:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

wdr ; Reset watchdog timer

Words: 1 (2 bytes)

Cycles: 1

Desactivar el Wattchdog

- Escribir un 1 en **WDTOE**
- Escribir un 1 en **WDE** (si esta en uno sobrescribir)
- En los siguientes 4 ciclos escribir un 0 en **WDE**

BITS DE SEGURIDAD PARA LAS MEMORIAS

SEGURIDAD DE LAS MEMORIAS FLASH Y EEPROM

- LOS ATmega164P POSEEN 6 BITS DE SEGURIDAD QUE PROTEGE A LAS MEMORIAS. SU CONFIGURACIÓN ES:
SIN PROGRAMAR = 1 LÓGICO
PROGRAMADOS = 0 LÓGICO
- VIENEN DESDE LA FÁBRICA SIN PROGRAMAR (TODOS EN 1 LÓGICO)
- TAMBIÉN QUEDAN SIN PROGRAMAR CUANDO SE ENVÍA EL COMANDO DE BORRAR AL CIRCUITO INTEGRADO

BYTE DE SEGURIDAD

BIT	Nº	DESCRIPCIÓN	VALOR POR DEFECTO
	7	-	1 (sin programar)
	6	-	1 (sin programar)
BLB12	5	BOOT LOCK BIT	1 (sin programar)
BLB11	4	BOOT LOCK BIT	1 (sin programar)
BLB02	3	BOOT LOCK BIT	1 (sin programar)
BLB01	2	BOOT LOCK BIT	1 (sin programar)
LB2	1	LOCK BIT	1 (sin programar)
LB1	0	LOCK BIT	1 (sin programar)

MODOS DE SEGURIDAD

MODO	LB2	LB1	TIPO DE PROTECCIÓN
1	1	1	Sin habilitar la protección de las memorias FLASH y EEPROM
2	1	0	Deshabilita futuras programaciones de la FLASH y la EEPROM, en forma Paralela o Serial (SPI y JTAG). Los bits de los Fusibles son asegurados en ambos modos
3	0	0	Igual al anterior y también se deshabilita la verificación (lectura)
Los bits de los Fusibles se programan antes que los de Seguridad			

SECCIONES DE LA FLASH

- LA MEMORIA DEL PROGRAMA SE DIVIDE EN DOS SECCIONES :
- LA PARTE BAJA PARA EL CÓDIGO DE LA APLICACIÓN, ESTÁ PROTEGIDA POR LOS BITS BLB01 y BL02
- LA PARTE ALTA PARA EL CÓDIGO DEL BOOT LOADER, ESTÁ PROTEGIDA POR LOS BITS BLB11 y BL12



SEGURIDAD DE LA APLICACIÓN

MODO	BLB02	BLB01	TIPO DE PROTECCIÓN
1	1	1	Sin restricciones en las instrucciones SPM o LPM, para el acceso a la sección de la Aplicación
2	1	0	SPM no está permitida para escribir en la sección de la Aplicación
3	0	0	Igual a los Modos 2 y 4 juntos
4	0	1	LPM ejecutada en el Boot Loader, no está permitida para leer desde la sección de la Aplicación. Si un vector de interrupción es puesto en la sección del Boot Loader, la interrupción está deshabilitada mientras se ejecuta desde la sección de la Aplicación

SEGURIDAD DEL BOOT LOADER

MODO	BLB12	BLB11	TIPO DE PROTECCIÓN
1	1	1	Sin restricciones en las instrucciones SPM o LPM, para el acceso a la sección del Boot Loader
2	1	0	SPM no está permitida para escribir en la sección del Boot Loader
3	0	0	Igual a los Modos 2 y 4 juntos
4	0	1	LPM ejecutada en la Aplicación, no está permitida para leer desde la sección del Boot Loader. Si un vector de interrupción es puesto en la sección de la Aplicación, la interrupción está deshabilitada mientras se ejecuta desde la sección del Boot Loader



FIN