Principios de Mecatrónica – SDI-11561
Ingeniería en Mecatrónica

Hugo Rodríguez Cortés

Departamento de Ingeniería Eléctrica y Electrónica

Instituto Tecnológico Autónomo de México

Agosto 2023

Operaciones Aritméticas

- MUL Rd, Rr Multiplicación de números sin signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- MULS Rd, Rr Multiplicación de números con signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- MULSU Rd, Rr Multiplicación de números con signo y números sin signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- Ejemplo

```
LDI   R23, 0x25   ;                              –
LDI   R24, 0x65   ;                              –
MUL   R23, R25    ;   0x25 × 0x65 = 0xE99–
                  ;   R1 = 0x0E, R0 = 0x99–
```

Operaciones Aritméticas

- En 0x310 se encuentra el dato 0xFD, convertirlo a decimal guardando en 0x322, 0x323 y 0x324 los dígitos; en 0x322 el menos significativo.

```
.EQU    HEX_NUM=0x315       ;                                   –
.EQU    RMND_L=0x322        ;                                   –
.EQU    RMND_M=0x323        ;                                   –
.EQU    RMND_H=0x324        ;                                   –
.DEF    NUM=R20             ;                                   –
.DEF    DENOMINATOR = R21   ;                                   –
.DEF    QUOTIENT = R22      ;                                   –

        LDI     R16, 0xFD           ;                           –
        STS     HEX_NUM, R16        ;                           –
        LDS     NUM, HEX_NUM        ;                           –
        LDI     DENOMINATOR, 10     ;                           –
        LDI     QUOTIENT, 0         ;                           –

L1:     INC     QUOTIENT            ;                           –
        SUB     NUM, DENOMINATOR    ;                           –
        BRCC    L1                  ;   salta si C=0–
        DEC     QUOTIENT            ;                           –
        ADD     NUM, DENOMINATOR    ;                           –
        STS     RMND_L, NUM         ;                           –
        MOV     NUM, QUOTIENT       ;                           –
        LDI     QUOTIENT, 0         ;                           –

L2:     INC     QUOTIENT            ;                           –
        SUB     NUM, DENOMINATOR    ;                           –
        BRCC    L2                  ;                           –
        DEC     QUOTIENT            ;                           –
        ADD     NUM, DENOMINATOR    ;                           –
        STS     RMND_M, NUM         ;                           –

        STS     RMND_H, QUOTIENT    ;                           –
```

Números con signo

- Para representar el signo de un número con signo se utiliza el bit más significativo

D7	D6	D5	D4	D3	D2	D2	D0
$\pm$	-	v	a	l	o	r	-

El rango de números positivos es de 0 a 127, si se requieren valores más grandes se deben utilizar operandos de 16 bits.

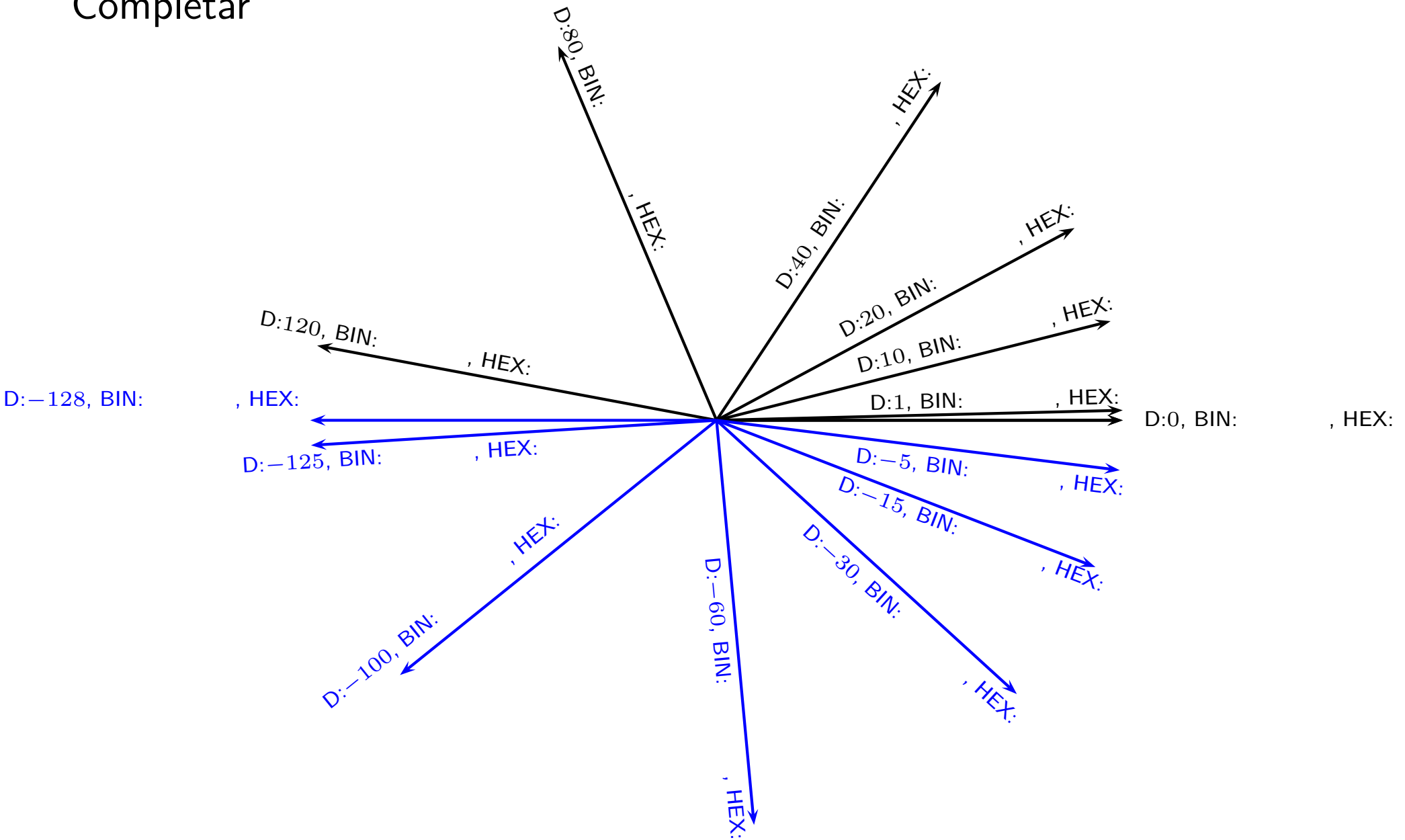
- En un número negativo D7 $= 1$ y el valor se representa con el complemento a 2.

- Ejemplo. Representar al número -5

0	0	0	0	0	1	0	1	5 en binario
1	1	1	1	1	0	1	0	complemento a 1
1	1	1	1	1	0	1	1	complemento a 2

Números con signo

Completar



Números con signo

- En operaciones con número con signo puede ocurrir un desbordamiento, el CPU indicará esta situación con la bandera V.

- Ejemplo.

```
LDI   R20, 0x60   ;   –
LDI   R21, 0x46   ;   –
ADD   R20, R21    ;   –
```

Números con signo

- En operaciones con número con signo puede ocurrir un desbordamiento, el CPU indicará esta situación con la bandera V.

- Ejemplo.

```
LDI   R20, 0x60   ;   –
LDI   R21, 0x46   ;   –
ADD   R20, R21    ;   –
```

$$0x60 + 0x46 = 0110\ 0000 + 0100\ 0110 = 1010\ 0110$$

El CPU indicará la situación con V=1. Notar que N=1 (resultado negativo).

- El rango de operación para números con signo son los números enteros $[-128,\ 127]$.

Números con signo

- En operaciones de 8 bits de números con signo la bandera V se asigna igual a 1 si alguna de las condiciones siguientes ocurre
  - Se lleva uno de D6 a D7 y no se lleva de D7 (C=0).
  - Se lleva uno de D7 (C=1) y no se lleva de D6 a D7
- Determine el valor de las banderas N y V en el siguiente código

```
LDI   R20, 0x80   ;   –
LDI   R21, 0xFE   ;   –
ADD   R20, R21    ;   –
```

Números con signo

- En operaciones de 8 bits de números con signo la bandera V se asigna igual a 1 si alguna de las condiciones siguientes ocurre
  - Se lleva uno de D6 a D7 y no se lleva de D7 (C=0).
  - Se lleva uno de D7 (C=1) y no se lleva de D6 a D7
- Determine el valor de las banderas N y V en el siguiente código

```
LDI   R20, 0x80   ;   –
LDI   R21, 0xFE   ;   –
ADD   R20, R21    ;   –
```

$$N = 0,\ V = 1$$

Números con signo

Examine el valor de las banderas V y N

- Sumar -2 con -5.

```
LDI   R20, -2     ;                    –
LDI   R21, -5     ;                    –
ADD   R20, R21    ;                    –
```

Números con signo

Examine el valor de las banderas V y N

- Sumar -2 con -5.

```
LDI   R20, -2    ;                    –
LDI   R21, -5    ;                    –
ADD   R20, R21   ;                    –
```

$$\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array} \begin{array}{l} \text{-2} \\ \text{-5} \\ \text{-7} \end{array}, \quad V = 0, N = 1$$

Números con signo

- Sumar 7 con 18.

```
LDI   R20, 7     ;   –
LDI   R21, 18    ;   –
ADD   R20, R21   ;   –
```

Números con signo

- Sumar 7 con 18.

```
LDI   R20, 7     ;                    –
LDI   R21, 18    ;                    –
ADD   R20, R21   ;                    –
```

$$\begin{array}{cccccccc|l} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 7 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 18 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 25 \end{array} , \quad V = 0, N = 0$$

- En operaciones sin signo C determina el estado de la operación, mientras que en operaciones con signo lo hace V.

Operaciones lógicas

- AND Rd, Rr Aplica el operador lógico AND bit a bit entre los registros Rd y Rr. ANDI Rd, k con k un valor constante.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI     R20, 0x35   ;                    –
ANDI    R20, 0x0F   ;                    –
```

Operaciones lógicas

- AND Rd, Rr Aplica el operador lógico AND bit a bit entre los registros Rd y Rr. ANDI Rd, k con k un valor constante.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI   R20, 0x35   ;                    –
ANDI  R20, 0x0F   ;                    –
```

$$\begin{array}{cccccccc} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \begin{array}{l} \text{0x35} \\ \text{0x0F} \\ \text{0x05} \end{array} , \quad Z = 0, N = 0$$

Operaciones lógicas

- OR Rd, Rr Aplica el operador lógico OR bit a bit entre los registros Rd y Rr. ORI Rd, k con k un valor constante.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI   R20, 0x04   ;                    –
ORI   R20, 0x30   ;                    –
```

Operaciones lógicas

- OR Rd, Rr Aplica el operador lógico OR bit a bit entre los registros Rd y Rr. ORI Rd, k con k un valor constante.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI   R20, 0x04   ;                    –
ORI   R20, 0x30   ;                    –
```

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array} \begin{array}{l} \text{0x04} \\ \text{0x30} \\ \text{0x34} \end{array} , \quad Z = 0, N = 0$$

Operaciones lógicas

- EXOR Rd, Rr Aplica el operador lógico XOR bit a bit entre los registros Rd y Rr.

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Esta operación puede activar las banderas Z, S y N.

```
LDI   R20, 0x54   ;                    –
LDI   R20, 0x78   ;                    –
EOR   R20, R21    ;                    –
```

Operaciones lógicas

- EXOR Rd, Rr Aplica el operador lógico XOR bit a bit entre los registros Rd y Rr.

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Esta operación puede activar las banderas Z, S y N.

```
LDI   R20, 0x54   ;                    –
LDI   R20, 0x78   ;                    –
EOR   R20, R21    ;                    –
```

$$\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{array} \begin{array}{l} \text{0x54} \\ \text{0x78} \\ \text{0x2C} \end{array} , \quad Z = 0, N = 0$$

Operaciones lógicas

- Analice el siguiente código

```
        LDI     R20, 0xFF     ;  –
        OUT     DDRC, R20     ;  –
        LDI     R20, 0x00     ;  –
        OUT     DDRB, R20     ;  –
        OUT     PORTC, R20    ;  –
        LDI     R21, 0x45     ;  –
HERE:
        IN      R20, PINB     ;  –
        EOR     R20, R21      ;  –
        BRNE    HERE          ;  –
        LDI     R20, 0x99     ;  –
        OUT     PORTC, R20    ;  –
```

Operaciones lógicas

- COM Rd Esta instrucción complementa bit a bit el contenido de un registro. Se conoce también como el complemento a uno.
- NEG Rd Esta instrucción calcula el complemento a dos de contenido del registro.
- CP Rd, Rr Esta instrucción es una resta entre Rd y Rr excepto que el valor de Rd no cambia.
- CPI Rd, k comparación con una constante. CP y CPI se unen a operadores de saltos condionales.

BCD y conversión ASCII

- BCD (binary coded decimal) es la representación binaria de los números del 0 al 9. Se asocia a dos conceptos BCD sin comprimir y BCD comprimido.

- En un BCD sin comprimir el nibble bajo representa al BCD y el nibble alto se llena con ceros. Por ejemplo, 9 = 00001001, 5 = 00000101. Se utiliza un byte de memoria para almacenarlo.

- En un BCD comprimido. Los nibbles alto y bajo contienen un número. Por ejemplo, 0x59 = 01011001. Se requiere la misma cantidad de memoria para almacenarlo, un byte

- En teclados ASCII cuando se activa la tecla '0' se envía 0110000 = 0x30 al CPU. De forma similar '2' = 0110010 = 0x32.

BCD y conversión ASCII

- Para convertir de BCD comprimido a ASCII. Primero el BCD se descomprime y luego se combina con 0x30.

BCD comprimido	BCD sin comprimir		ASCII	
0x29	0x02	0x09	0x32	0x39
0010 1001	0000 0010	0000 1001	0011 0010	0011 1001

- Código

```
LDI   R20, 0x29   ;                          –
MOV   R21, R20    ;                          –
ANDI  R21, 0x0F   ;                          –
ORI   R21, 0x30   ;                          –

MOV   R22, R20    ;                          –
SWAP  R22         ;  intercambia nibbles–
ANDI  R22, 0x0F   ;                          –
ORI   R22, 0x30   ;                          –
```

BCD y conversión ASCII

- Para convertir de ASCII a BCD comprimido. Primero el BCD se descomprime y luego se combina para comprimirlo.

Tecla	ASCII	BCD sin comprimir	BCD comprimido
'4'	0x34	00000100	
'7'	0x37	00000111	01000111=0x47

- Código

```
LDI     R21, '4'     ;                    –
LDI     R22, '7'     ;                    –
ANDI    R21, 0x0F    ;                    –
SWAP    R21          ;                    –

ANDI    R22, 0x0F    ;                    –
OR      R22, R21     ;                    –
MOV     R20, R22     ;                    –
```

Operaciones Ar/Lo con constantes

- El CPU acepta operaciones aritméticas del tipo

```
.EQU   ALFA = 50   ;                          –
.EQU   BETA = 40   ;                          –

  LDI   R23, ALFA                  ;                    –
  STS   R24, ((ALFA-BETA)*2)+9     ;                    –
```

- Lo mismo que operaciones lógicas como

```
.EQU   C1 = 0x50   ;                          –
.EQU   C2 = 0x10   ;                          –
.EQU   C3 = 0x04   ;                          –

  LDI   R21, (C1&C2)|C3   ;                        –
```

Operaciones Ar/Lo con constantes

- Operaciones aritméticas

Símbolo	Acción
$+$	Adición
$-$	Resta
$*$	Multiplicación
$/$	División
$\%$	Módulo

- Operaciones lógicas

Símbolo	Acción
$\&$	AND bit a bit
$\mid$	OR bit a bit
$\wedge$	XOR bit a bit
$\sim$	NOT bit a bite

Operaciones Ar/Lo con constantes

- Operadores de corrimiento

Símbolo	Acción	Ejemplo
X<<Y	Recorre hacia la izquerda a X	LDI R20, 0b101<<2
	Y número de lugares y viceversa	;R20 = 0b10100
X>>Y	Recorre hacia la derecha a X	LDI R20, 0b100>>1
	Y número de lugares y viceversa	;R20 = 0b010

- Ejemplo

```
LDI   R21, 0b00000111<<1   ;   R21 = 0b00001110-
```

- HIGH() y LOW() Obtienen el byte alto y bajo, respectivamente de un dato de 16 bits

```
LDI   R21, LOW(0x4455)   ;   R21 = 0x55-
LDI   R22, HIGH(0x4455)  ;   R22 = 0x44-
```

Timers

- Muchas aplicaciones necesitan contar eventos y/o generar tiempos de retardo. El microcontrolador tiene **registros de contadores** para estos propósitos.

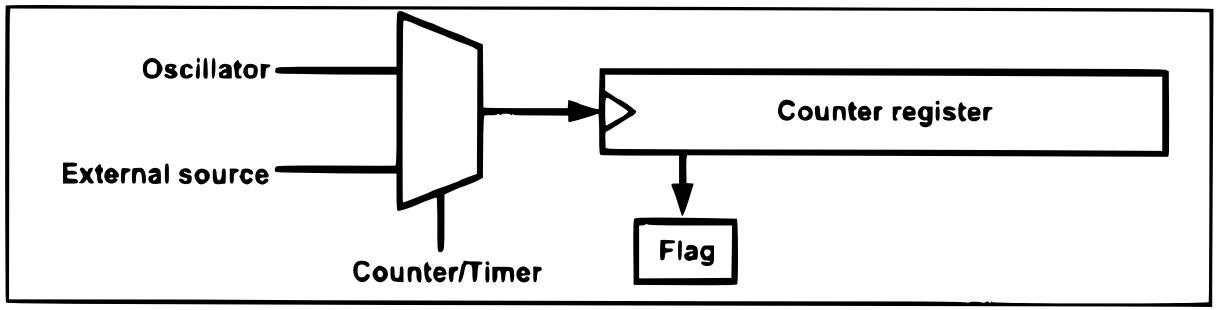
- Para contar un evento se conecta la fuente del evento externo al pin del registro del contador. De esta forma, cuando el evento ocurre el contador se incremeta, así, el contenido del contador representa el número de veces que el evento ocurre.

- Para generar tiempo de retardo, se conecta el oscilador al pin del contador. Cada oscilación incrementa el contenido del contador. Como resultado, el contenido del registro del contador representará cuantas oscilaciones han ocurrido desde que limpió el contador.

Timers

- Ya que la velocidad del oscilador, en un microcontrolador, se conoce, se puede calcular el periodo de la oscilación y del contenido del registro del contador se obtiene el tiempo transcurrido.

- De esta manera para generar un retardo, se borra el contador en el tiempo inicial y se espera hasta que el contador alcance un cierto número.



Timers

- Ejemplo, considere un microcontrolador con un oscilador de 1MHz; en el microcontrolador, el contenido del registro del contador incrementa una vez por cada microsegundo. Entonces, si se quiere un retardo de 100 microsegundos, debemos limpiar el contador y esperar hasta que sea igual a 100.

- En el microcontrolador, hay una bandera para cada uno de los contadores.
  - La bandera se activa cuando el contador se desborda, por software se borra.
  - El segundo método para generar un retardo es cargar el registro del contador y esperar hasta que el contador se desborde y la bandera se active.

Timers

- Ejemplo. En un microcontrolador, con oscilador de 1MHz y registro de contador de 8 bits. Si se requiere un retardo de 3 microsegundos, el registro del contador se carga con 0xFD, la bandera se activará después de tres ciclos. El contenido del registro se incrementara a 0xFE, después a 0xFF y en el tercer ciclo el contenido del registro se desborda a 0x00 y la bandera se activa.

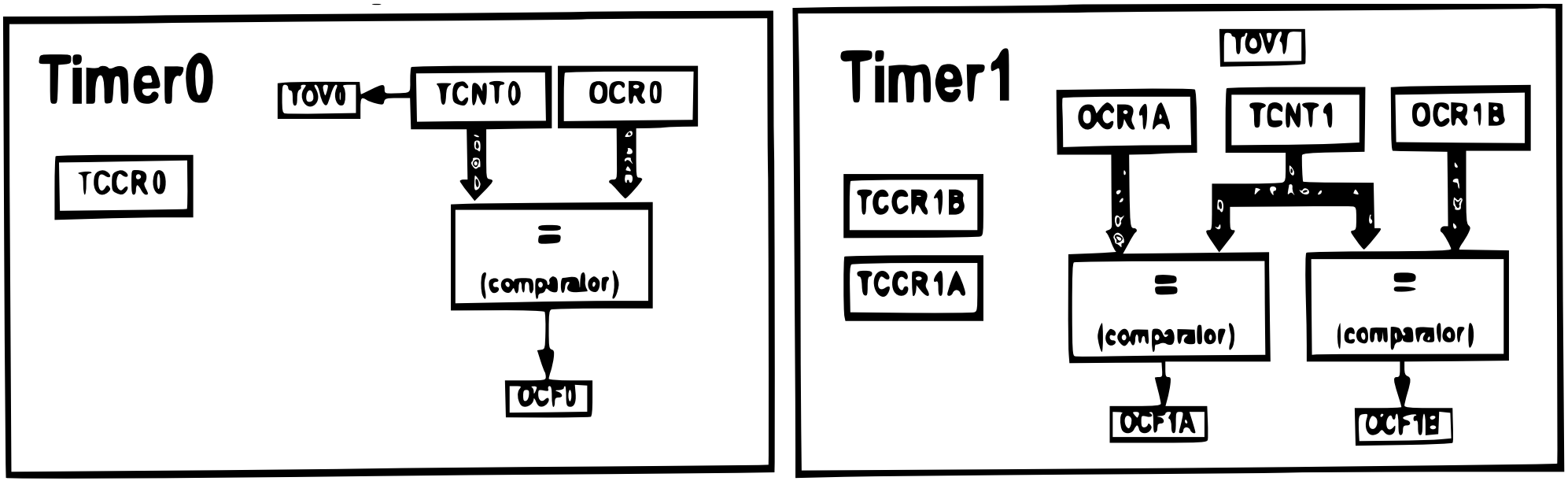
- El AVR tiene de 1 a 6 timers dependiendo del tipo de familia. Ellos son referidos como 0, 1..5. Pueden ser usados como timers para generar retardo o como contadores para contar eventos sucediendo fuera del microcontrolador.

- El ATmega32, tiene 3 timers: Timer0, Timer1 y Timer2. Timer0 y Timer2 son de 8 bits, mientras que el timer T1 es de 16 bits.

Timers

- Cada Timer necesita un pulso de reloj para contar. La fuente del pulso puede ser interna o externa.
- Si se usa una fuente interna, entonces la frecuencia del oscilador alimenta al contador. Por lo tanto, al utilizarse para generar retardos se conoce como Timer.
- Si se elige la opción de un pulso externo, es necesario alimentarlo a través de uno de los pines del AVR. En este caso se tiene un contador.

AVR Timer



TCNTn (timer/counter) registro del Timer/contador. TOVn (Timer Overflow) bandera de desbordamiento. TCCRn (timer/counter control register) registro para definir modos de operación. OCRn (Output Compare Register). El contenido de OCRn se compara con TCNTn, si son iguales la bandra OCFn (Output Compare Flag) se activa.

Programación Timer0

- TCNT0 El registro Timer/Contador tiene 8 bits

TCNT0 D7 D6 D5 D4 D3 D2 D1 D0

- TCCR0 Control/Modo de Operación del registro TCNT0

TCCR0	D7	D6	D5	D4	D3	D2	D1	D0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
	W	RW	RW	RW	RW	RW	RW	RW
CI	0	0	0	0	0	0	0	0

FOC0: Se usa para generar ondas. WGM00-WGM01: Modo de Operación 0-0 normal, 0-1 Limpiar el Timer al comparar, 1-0 PWM fase correcta, 1-1 PWM rápido. COM00-COM01: Se utilizan en la generación de ondas. CS02-CS01-CS00: Configuración del reloj, 0-0-0 Timer/Contador detenido, 0-0-1 Reloj clk sin escalador, 0-1-0 clk/8, 0-1-1 clk/64, 1-0-0 clk/256, 1-0-1 clk/1024. 1-1-0 reloj en pin T0 flanco de bajada, 1-1-1 reloj en pin T0 flanco de subida.

Programación Timer0

- TIFR0 Registro de la bandera de interrupción del registro del Timer/Contador.

TIFR0	D7	D6	D5	D4	D3	D2	D1	D0
	OFC2	TOV2	ICF1	OFC1A	OFC1B	TOV1	OFC0	TOV0
	RW	RW	RW	RW	RW	RW	RW	RW
CI	0	0	0	0	0	0	0	0

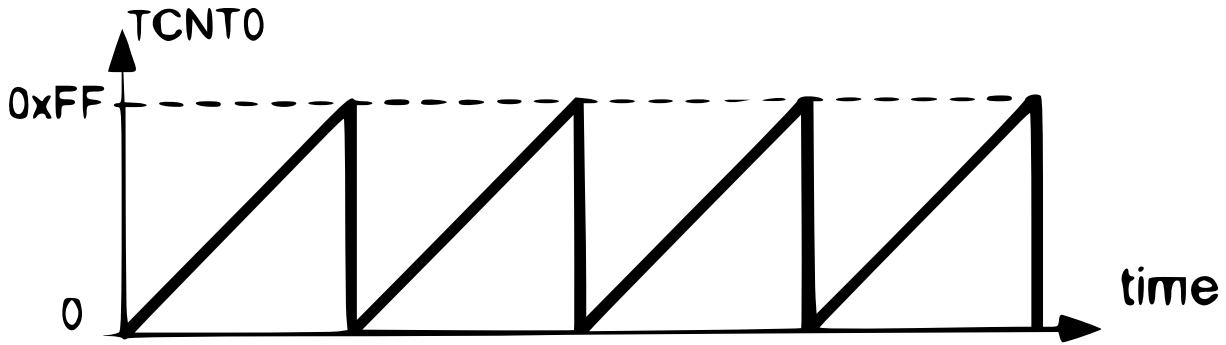
TOV0: 0-1 no desbordamiento/desbordamiento Timer0, 0xFF→0x00. OFC0: 0-1 comparación ocurrió/no ocurrió. TOV1: 0-1 no desbordamiento/desbordamiento Timer1. OFC1B: Bandera de comparación de salida B del Timer1. OFC1A: Banderade comparación de salida A del Timer1. ICF1: Bandera de captura de entrada. TOV2: Bandera de desbordamiento Timer2. OFC2: Bandera de comparación de salida.

- La bandera TOV0 es igual a uno cuando TCNT0 se desborda, decir va de 0xFF a 0x00. Para borrar la bandera se escribe uno en TOV.

```
LDI    R21, 0x01    ;                    –
OUT    TIFR, R21    ;                    –
```

Programación Timer0

- En este modo, el contenido del timer/contador aumenta cada ciclo de reloj. El contador se incrementa hasta alcanzar su máximo 0xFF, cuando él pasa de 0xFF a 0x00, la bandera TVO0 se activa. Esta bandera se puede leerse.



Programación Timer0

```
        LDI     R16, 1<<5           ;                                   –
        SBI     DDRB, 5             ;                                   –
        LDI     R17, 0x00           ;                                   –
        OUT     PORTB, R17          ;                                   –

BEGIN:  RCALL   DELAY               ;                                   –
        EOR     R16, R17            ;                                   –
        OUT     PORTB, R17          ;                                   –
        RJMP    BEGIN               ;                                   –

DELAY:  LDI     R20, 0xF2           ;                                   –
        OUT     TCNT0, R20          ;                                   –
        LDI     R20, 0x01           ;                                   –
        OUT     TCCR0, R20          ;                                   –

AGAIN:  IN      R20, TIFR           ;                                   –
        SBRS    R20, TOV0           ;  SBRS Verifica el estatus de un bit, salta si TVO0=1–
        RJMP    AGAIN               ;                                   –

        LDI     R20, 0x00           ;                                   –
        OUT     TCCR0, R20          ;                                   –
        LDI     R20, (1<<TOV0)      ;                                   –
        OUT     TIFR, R20           ;                                   –
        RET                         ;                                   –
```

Programación Timer0

- ¿Cuál es el valor del retardo generado por el timer asumiendo que el cristal(XTAL) es de 8 MHZ?

Programación Timer0

- ¿Cuál es el valor del retardo generado por el timer asumiendo que el cristal(XTAL) es de 8 MHZ?

  - La frecuencia del reloj que alimenta al Timer es de 8MHz, entonces cada ciclo de reloj tiene un periodo de T= $1/(8MHz)$ =0.125 micro segundos. Esto significa que el Timer0 cuenta hacia arriba cada 0.125 microsegundos, esto es,

    Delay=número de cuentas x (0.125 microsegundos).

  - El contador inicia en 0xF2 y la bandera TOV0 se activa al llegar a 0x00, entonces, sumaríamos un ciclo extra (0xFF-0xF2=13+1). Así, tendremos que el retardo es de 1.75 microsegundos.

Programación Timer0

Ejemplo. Suponga que se tiene un cristal de 8MHz. Modificar el programa anterior para generar una onda cuadrada con periodo de 12.5 μs en el pin 3 del PORTB.

Programación Timer0

Ejemplo. Suponga que se tiene un cristal de 8MHz. Modificar el programa anterior para generar una onda cuadrada con periodo de 12.5 μs en el pin 3 del PORTB.

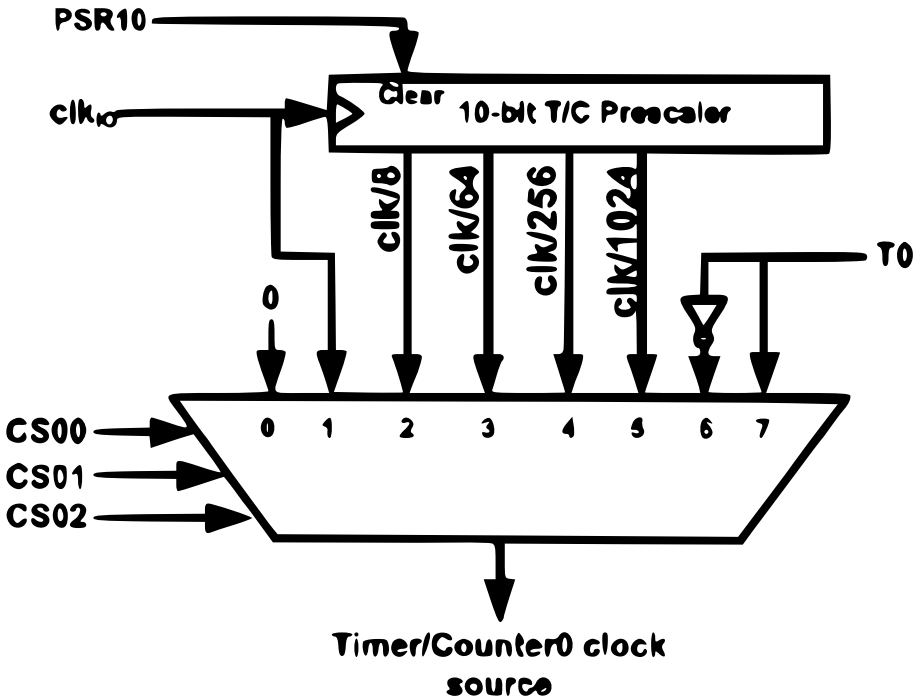
Para generar la onda cuadrada con T $= 12.5\mu$s se requiere un retardo de 6.25μs. Con el cristal de 8MHz se tiene un contador de 0.125 μs.
Por lo tanto, se requieren 6.25/0.125 ciclos. Las modificaciones al programa anterior son las siguientes

```
        LDI     R16, 0x08       ;                       –
        SBI     DDRB, 3         ;                       –
        LDI     R17, 0x00       ;                       –
        OUT     PORTB, R17      ;                       –

DELAY:  LDI     R20, 0xCE       ;                       –
```

Programación Timer0

- El control del retardo recae en el resgistro de 8 bits TNCT0 y la frecuencia del Cristal XTAL.

- El AVR tiene la posibilidad de escalar la frecuencia del cristal por un factor de 8, 64, 256 y 1024.



Programación Timer0

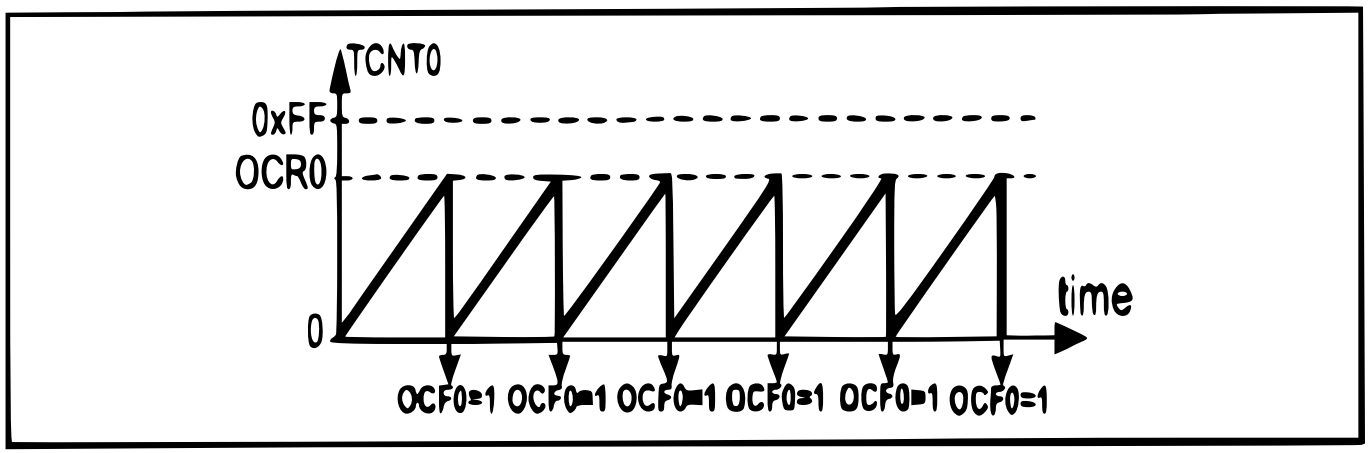
Calcule el retardo generado por el siguiente programa con XTAL=8MHz

```
        LDI     R16, 0x08           ;  –
        SBI     DDRB, 3             ;  –
        LDI     R17, 0x00           ;  –
        OUT     PORTB, R17          ;  –

BEGIN:  RCALL   DELAY               ;  –
        EOR     R16, R17            ;  –
        OUT     PORTB, R17          ;  –
        RJMP    BEGIN               ;  –

DELAY:  LDI     R20, 0x00           ;  –
        OUT     TCNT0, R20          ;  –
        LDI     R20, 0x05           ;  –
        OUT     TCCR0, R20          ;  –

AGAIN:  IN      R20, TIFR           ;  –
        SBRS    R20, TOV0           ;  SBRS Verifica el estatus de un bit, salta si TVO0=1–
        RJMP    AGAIN               ;  –

        LDI     R20, 0x00           ;  –
        OUT     TCCR0, R20          ;  –
        LDI     R20, (1<<TOV0)      ;  –
        OUT     TIFR, R20           ;  –
        RET                         ;  –
```

Programación Timer0

- El registro OCR0 se utiliza con el modo CTC. Al igual que en el modo normal, en el modo CTC, el timer se incrementa con el reloj, pero ahora cuenta hasta que el contenido del registro TCNT0 sea igual al contenido de OCR0.

- En esta punto la bandera OCF0 se activa en el siguiente ciclo de reloj. La bandera OCF0 esta localiza en el registro TIRF.



Programación Timer0

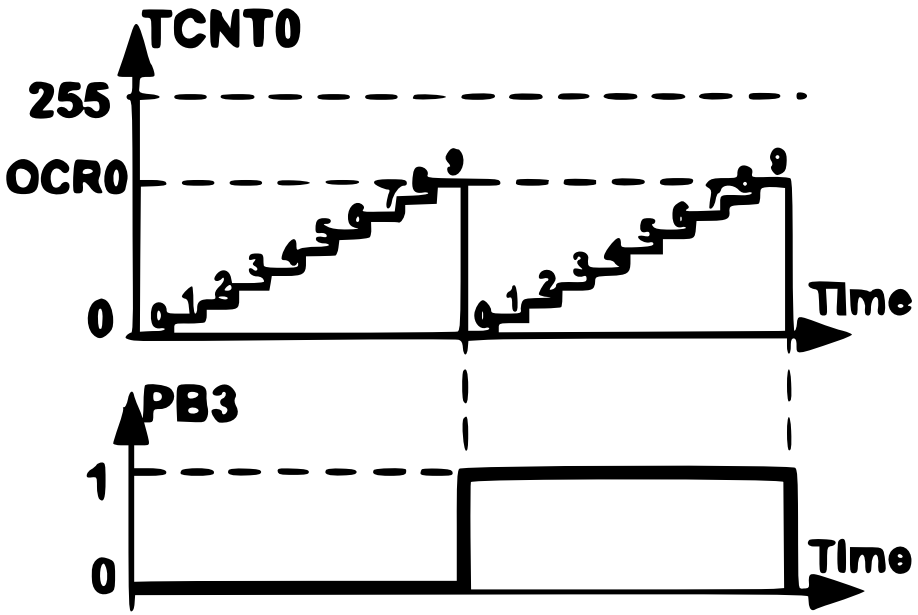
Determine la salida generada por el siguiente programa

```
        LDI     R16, 0x08       ;   –
        SBI     DDRB, 3         ;   –
        LDI     R17, 0x00       ;   –
        OUT     PORTB, R17      ;   –

BEGIN:  OUT     PORTB, R17      ;   –
        RCALL   DELAY, R17      ;   –
        EOR     R17, R16        ;   –
        RJMP    BEGIN           ;   –

DELAY:  LDI     R20, 0x00       ;   –
        OUT     TCNT0, R20      ;   –
        LDI     R20, 9          ;   –
        OUT     OCR0, R20       ;   –
        LDI     R20, 0x09       ;   –
        OUT     TCCR0, R20      ;   –
AGAIN:  IN      R20, TIFR       ;   –
        SBRS    R20, OCFR0      ;   –
        RJMP    AGAIN           ;   –
        LDI     R20, 0x00       ;   –
        OUT     TCCR0, R20      ;   –
        LDI     R20, 1<<OCF0    ;   –
        OUT     TIFR, R20       ;   –
        RET                     ;   –
```

Programación Timer0

- OCR0 se carga con 9 y TCNT0 con 0. Después de 9 pulsos de reloj TCNT0=OCR0. En el siguiente ciclo, la bandera OCF0 es igual a 1 y ocurre el reset. Esto significa que TCNT0 se limpia después de 10 ciclos.

- Ya que XTAL= 8MHz, el contador cuenta cada 0.125 μs. Por lo tanto, tenemos que $10 \times 0.125\mu s =$ 1.25μs es el valor del retardo.



Programación Timer0 en ATMega328P

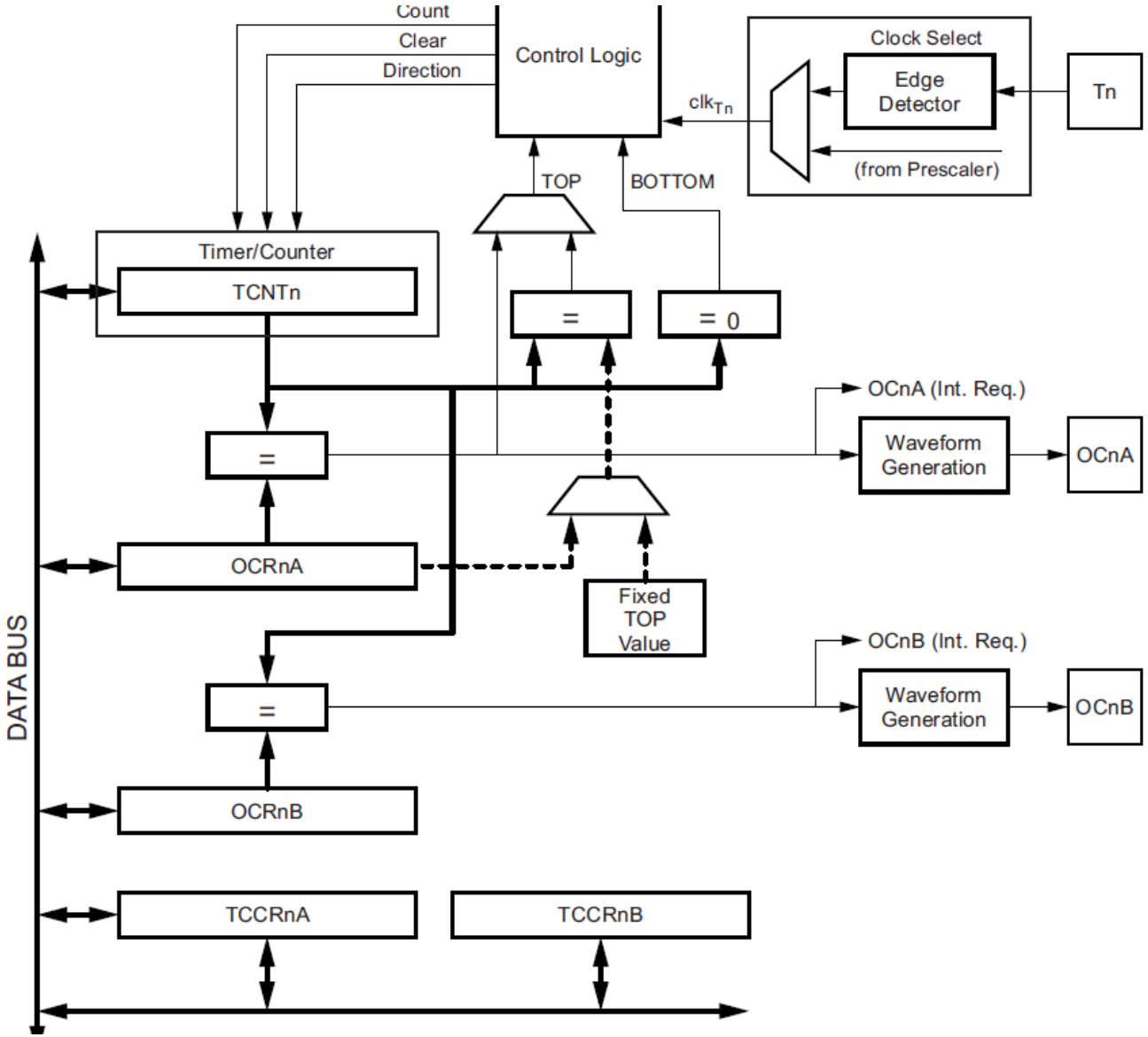
Es siempre recomendable recurrir al manual del microcontrolador, el Timer0 en el AVR ATMega328P es ligeramente diferente, Capítulo 14.

8-bit Timer/Counter0 with PWM

Features

- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

Programación Timer0 en ATMega328P



Programación Timer0 en ATMega328P

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
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	

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	

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Programación Timer2

Aunque los Timer0 y Timer2 son de 8 bits, existen dos diferencias entre ellos.

- El Timer2 puede utilizarse como un contador de tiempo real conectando un cristal extra de 32.768KHz en los pines TOSC1 y TOSC2.

- Los bits en el registro TCCR tienen diferente significado.

Programación Timer1

- El Timer1 es de 16 bits, sus registros se dividen en dos bytes. Para manejar 16 bits, el AVR utiliza una localidad de memoria temporal por lo que deben escribirse o leerse juntos, primero el byte alto.[a]

```
LDI   R20, 0x08      ;   –
LDI   R21, 0x00      ;   –
STS   TCNT1H, R20    ;   –
STS   TCNT1L, R21    ;   –
```

- Registro del Timer1/Contador1 TCNT1L-TCNT1H Bytes bajo y alto.

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
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	

[a] En el ATMega328P los registros TCNT1H-TCNT1L se encuentran en la memoria I/O por lo que debe utilizarse STS para cargar datos.

Programación Timer1

- TCCR1A-TCCR1B-TCCR1C Registros de control del Timer1/Contador1.

TCCR1A – Timer/Counter1 Control Register A

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

TCCR1B – Timer/Counter1 Control Register B

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

TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Programación Timer1

- OCCR1A-OCCR1B. Los registros OCCR están compuestos de dos bytes. Por ejemplo, OCCR1A esta compuesto de OCCR1AH (byte alto) y OCCR1AL(byte bajo).

OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
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	

OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
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	

Programación Timer1

- TIFR1 Registro de banderas de interrupción del Timer1/Contador1

TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Programación Timer1

Determine la salida generada por el siguiente programa

```
        LDI     R16, 0x20       ;   –
        SBI     DDRB, 5         ;   –
        LDI     R17, 0x00       ;   –
        OUT     PORTB, R17      ;   –

BEGIN:  RCALL   DELAY           ;   –
        EOR     R17, R16        ;   –
        OUT     PORTB, R17      ;   –
        RJMP    BEGIN           ;   –

DELAY:  LDI     R20, 0xD8       ;   –
        LDI     R21, 0xF0       ;   –
        STS     TCNT1H, R20     ;   –
        STS     TCNT1L, R21     ;   –
        LDI     R20, 0x00       ;   –
        LDI     R21, 0x05       ;   –
        STS     TCCR1A, R20     ;   –
        STS     TCCR1B, R21     ;   –
AGAIN:  IN      R20, TIFR1      ;   –
        SBRS    R20, TOV1       ;   –
        RJMP    AGAIN           ;   –
        LDI     R20, 0x00       ;   –
        STS     TCCR1A, R20     ;   –
        STS     TCCR1B, R20     ;   –
        LDI     R20, 0x01       ;   –
        OUT     TIFR1, R20      ;   –
        RET                     ;   –
```

Encontrando valores para ser cargados en el timer

Asumiendo que se conoce el valor del retardo que necesitamos; la pregunta sería: ¿Cómo encontrar los valores necesarios para el registro TCNT0/1? Para calcular los valores a ser cargados en el registro TCNT0/1; podemos seguir los siguientes pasos:

- Calcular el período del reloj usando la formula:

$$T_{clock} = 1/(K \times F_{timer})$$

  dónde F_{timer} es la frecuencia del reloj usado y K es el factor de escalamiento. Por ejemplo, en modo no escalado $K = 1$, $F_{timer} = F_{frec.\ oscilador}$. T_{clock} es el periodo en el cual el timer se incrementa.

- Dividir el retardo deseado por T_{clock}. Esto determinará cuantos pulsos de reloj son necesarios.

- Realizar $256/65536 - n$, donde n es el número decimal que se obtuvo del paso anterior.

- Convertir el número decimal del paso anterior a formato hexadecimal, siendo este valor el que se cargará en el registro del timer0/1.

Programación Contador0/1

- Los Timers del AVR pueden además utilizarse para contar, detectar y medir el tiempo de eventos pasando fuera del AVR.
- Cuando el Timer se usa para generar retardos, el cristal del AVR es la fuente de frecuencia.
- Cuando se usa como contador, es un pulso externo el que incrementa el registro TCNTx.
- Recordar que los bits CS02-CS01-CS00 del registro TCCR0B/TCCR1B decide la fuente del reloj para el timer. Si CS02-CS01-CS00 = 001-101 el Timer0/1 obtiene pulsos del cristal oscilador, posiblemente con escalamiento. En contraste, cuando CS02-CS01-CS00 = 110-111 el Timer0/1 se usa como un contador y obtiene sus pulsos de una fuente externa al AVR.

Programación Contador0/1

- Por lo tanto, cuando CS02-CS01-CS00 = 110-111 el registro del contador TCNT0/TCNT1 cuenta hacia arriba (pulso subida o de bajada) cada que llegan pulsos al pin T0/T1.
- Ejemplo

```
        CBI     DDRB, 0          ;  –
        LDI     R20, 0xFF        ;  –
        OUT     DDRC, R20        ;  –
        LDI     R20, 0X06        ;  –
        OUT     TCCR0, R20       ;  –

AGAIN:  IN      R20, TCNT0       ;  –
        OUT     PORTC, R20       ;  –
        IN      R16, TIFR0       ;  –
        SBRS    R16, TOV0        ;  –
        RJMP    AGAIN            ;  –
        LDI     R16, 1<<TOV0     ;  –
        OUT     TIFR0, R16       ;  –
        RJMP    AGAIN            ;  –
```

Programación Contador0/1

- Ejemplo, contar hasta un número de 16 bits.

```
        LDI     R19, 0          ;   –
        CBI     DDRB, 0         ;   –
        LDI     R20, 0xFF       ;   –
        OUT     DDRC, R20       ;   –
        OUT     DDRD, R20       ;   –
        LDI     R20, 0x06       ;   –
        OUT     TCCR0, R20      ;   –

AGAIN:  IN      R20, TCNT0      ;   –
        OUT     PORTC, R20      ;   –
        IN      R16, TIFR0      ;   –
        SBRS    R16, TOV0       ;   –
        RJMP    AGAIN           ;   –
        LDI     R16, 1<<TOV0    ;   –
        OUT     TIFR0, R16      ;   –
        INC     R19             ;   –
        OUT     PORTD, R19      ;   –
        RJMP    AGAIN           ;   –
```

Programación Timers

- ¿Qué hace el CPU durante la implementación del retardo o del contador?