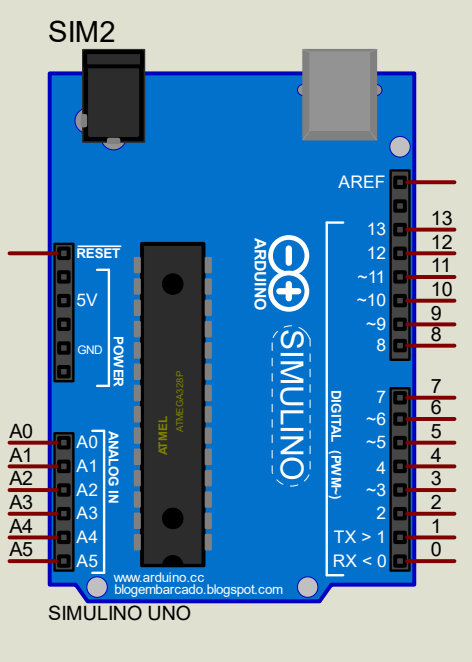
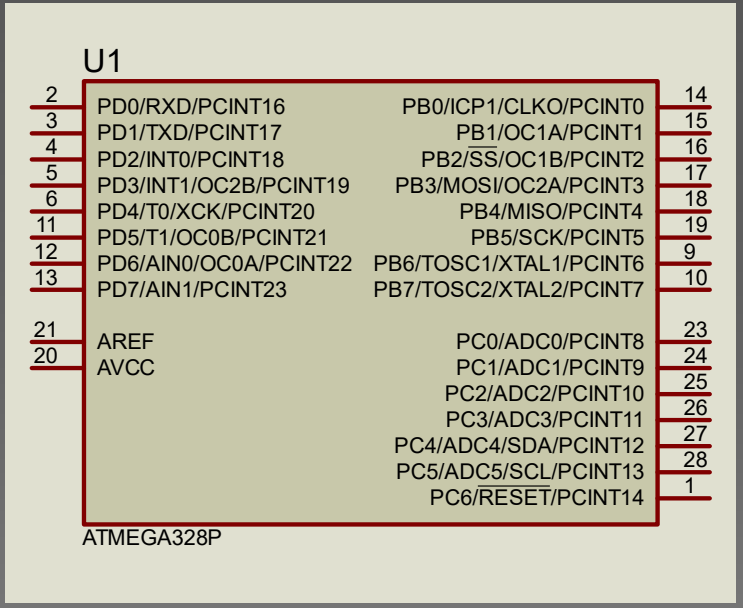
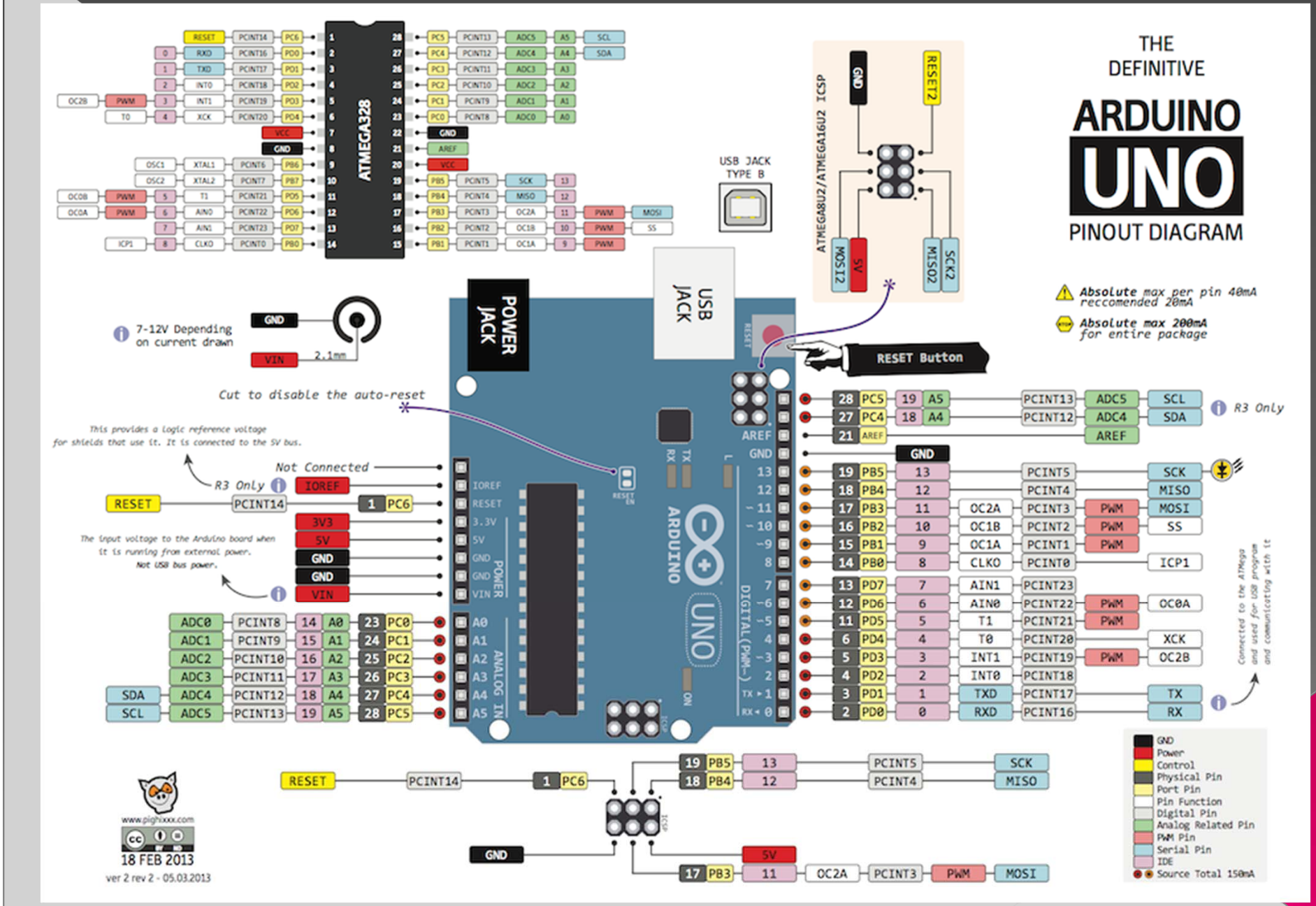
AVR – ATMEGA328p

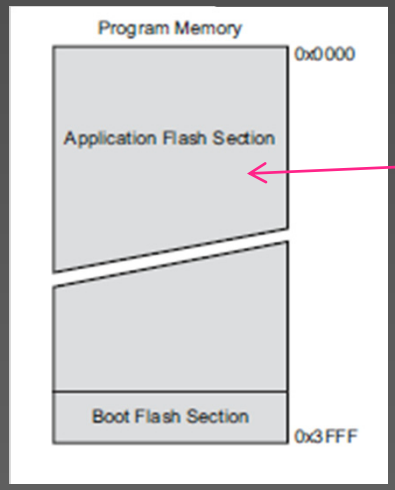




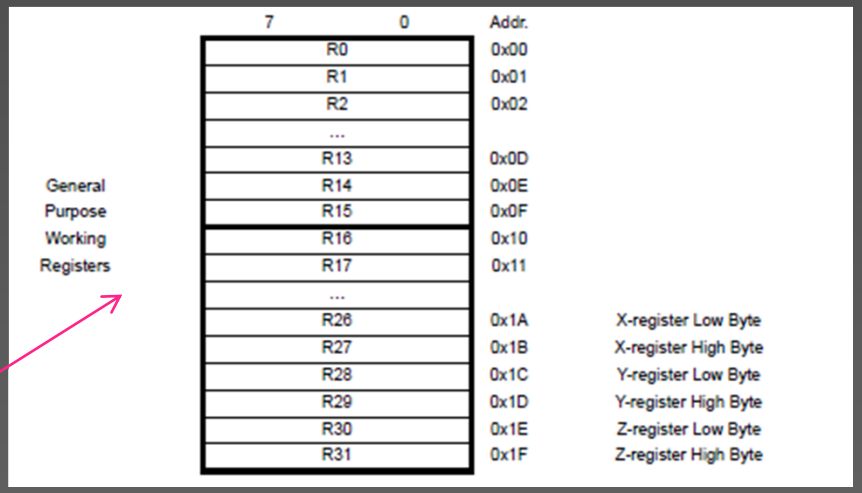
Circuitos Digitales y Microprocesadores
Arquitectura de Compurtadoras I
FI - UNLP

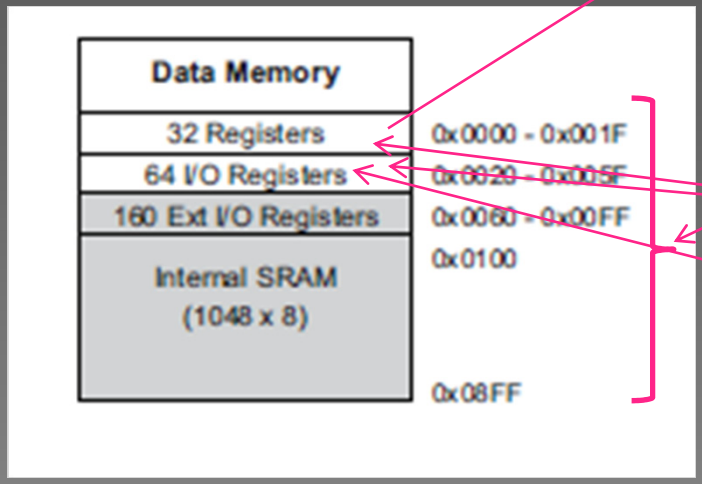

THE
DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK
TYPE B
ATMEGA8U2/ATMEGA16U2 ICSP
RESET2
MOSI2
5V
MISO2
SCK2
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
POWER
JACK
USB
JACK
RESET Button
7-12V Depending
on current drawn
2.1mm
Cut to disable the auto-reset
This provides a logic reference voltage
for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when
it is running from external power.
Not USB bus power.
Connected to the ATMega
and used for USB program
and communicating with it
ARDUINO
UNO
DIGITAL(PWM~)
POWER
ANALOG IN
ICSP
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013

Mapa de Memoria



.db,.dw
LPM,SPM
(Z)



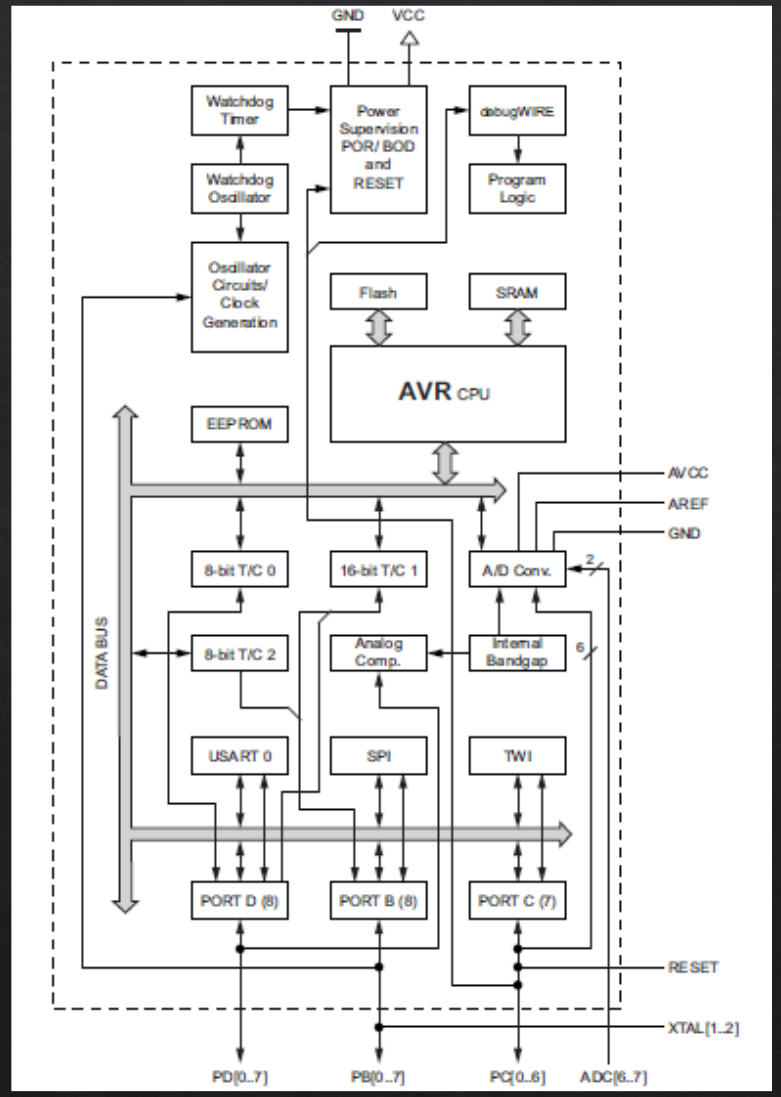


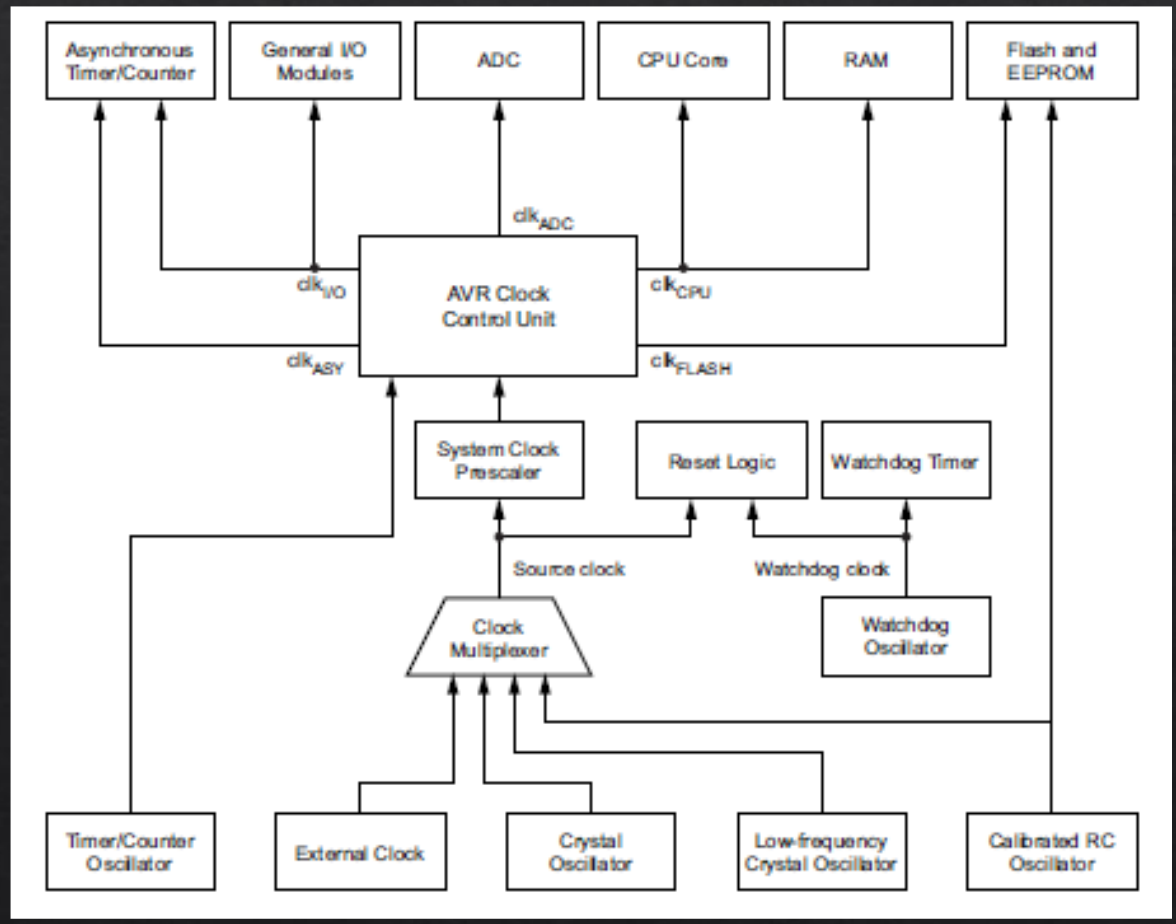
LD,LDS,LDD
ST,STS,STD

SBI,CBI,SBIS,SBIC

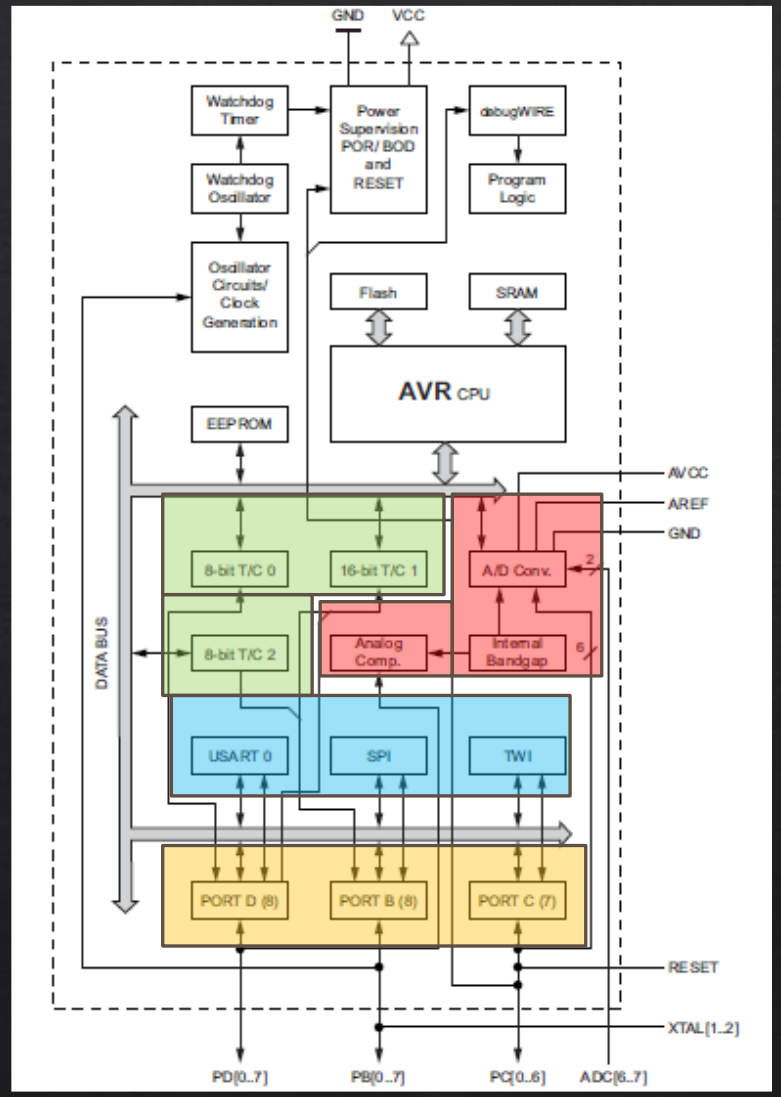
IN,OUT (0x0000 – 0x003F)

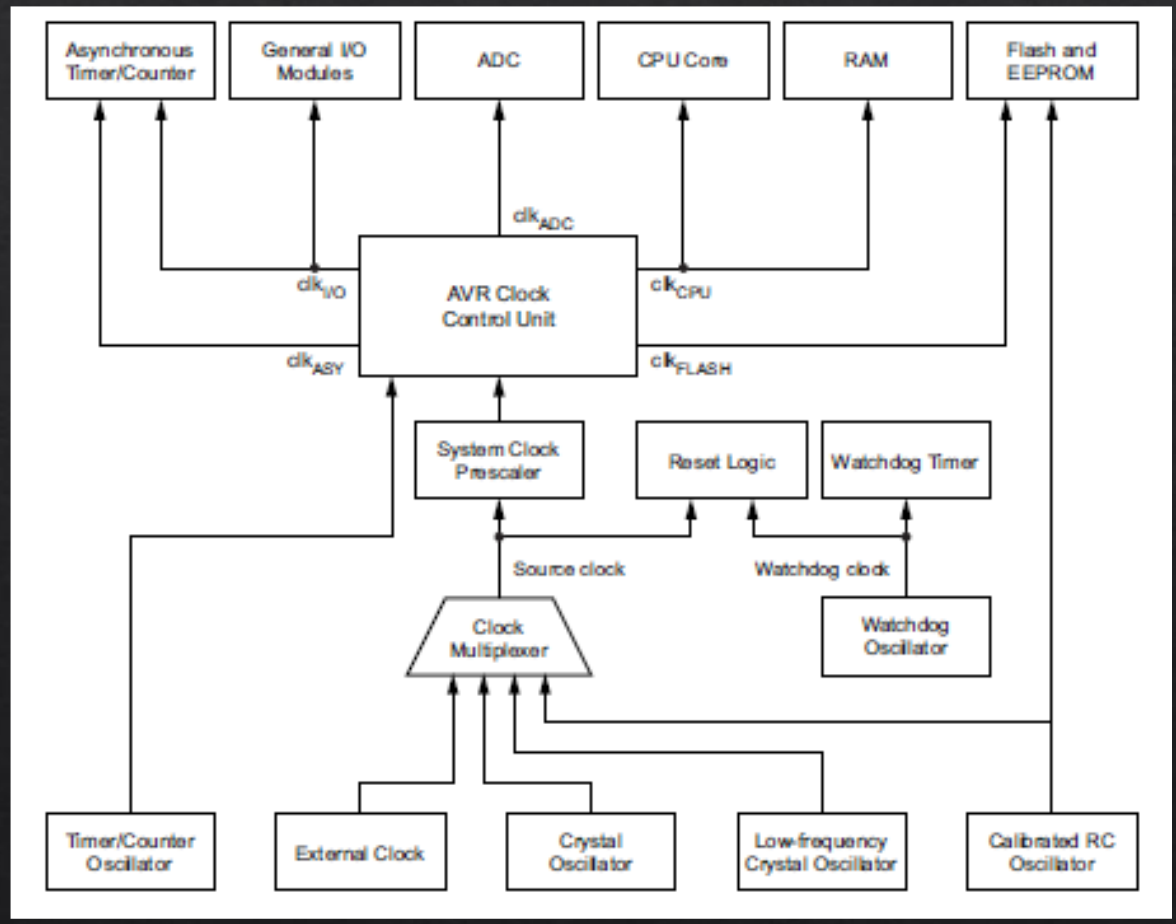
AVR ATmega328p



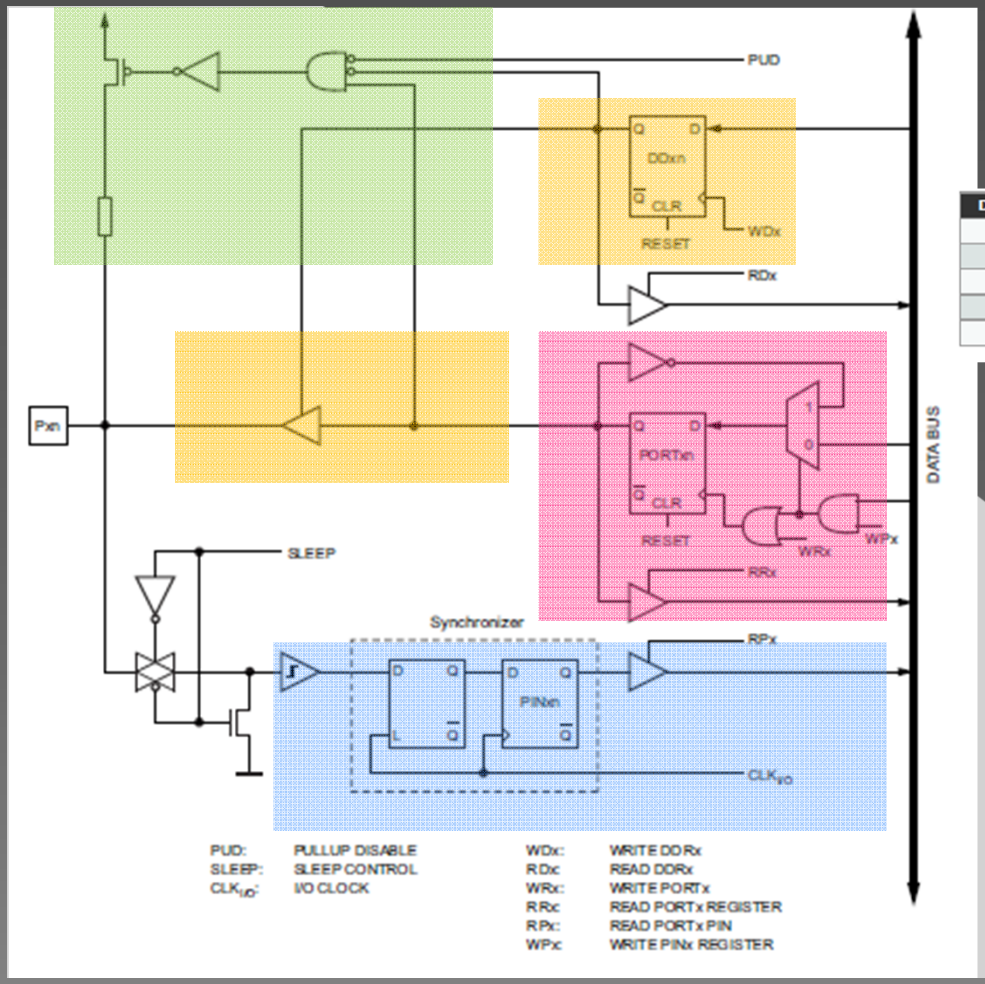


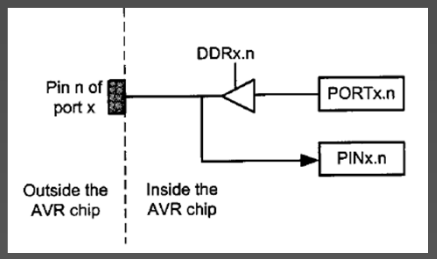
AVR ATmega328p

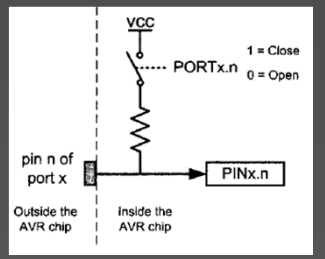




Puertos de E/S







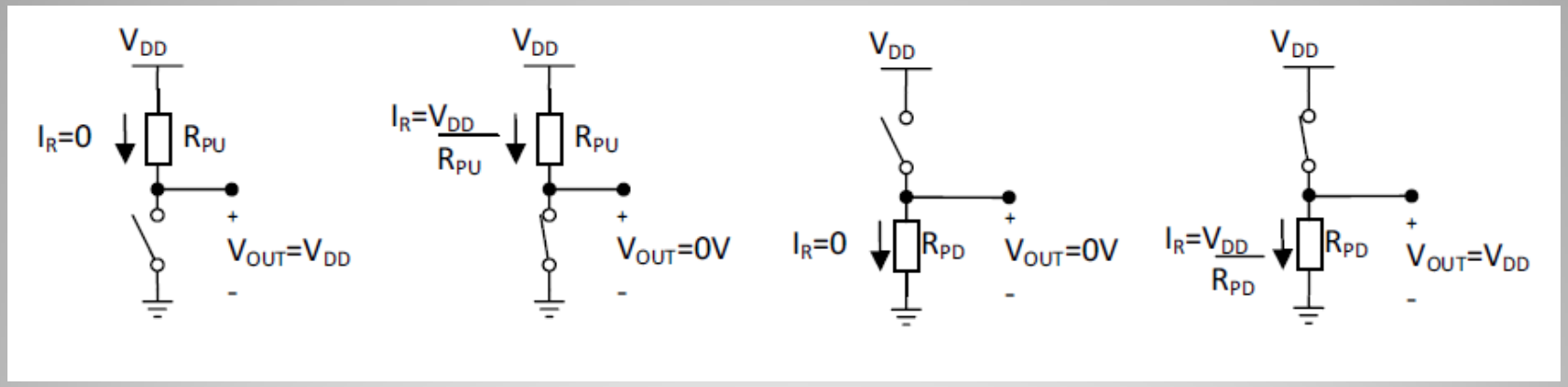
DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output low (sink)
1	1	X	Output	No	Output high (source)

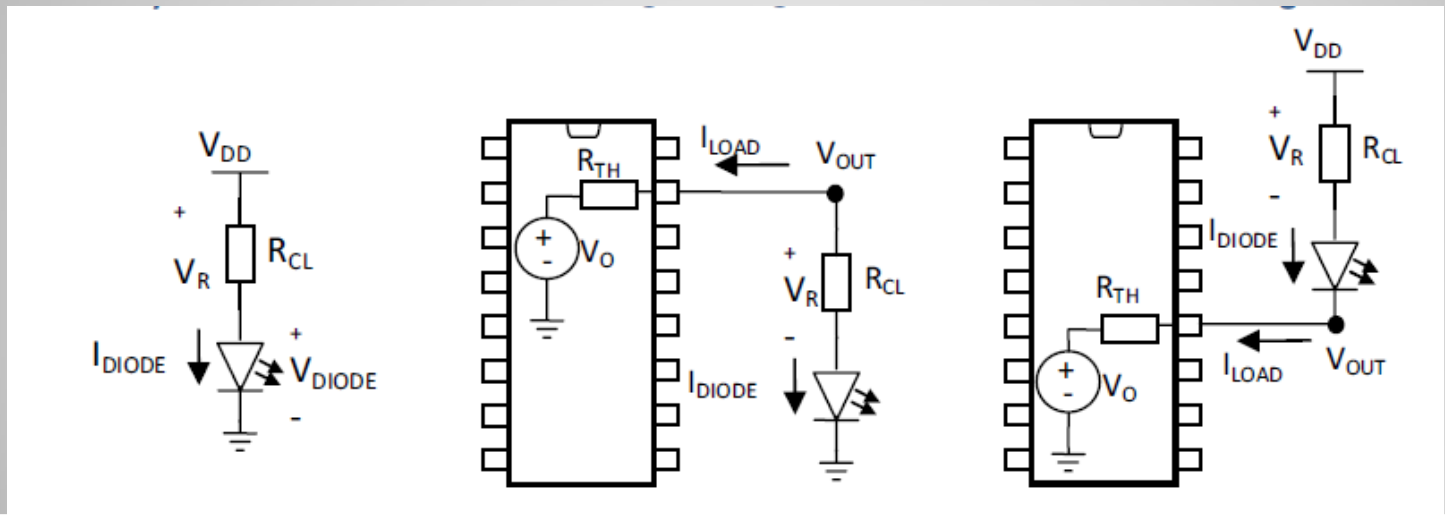
- Configuración de Dirección E/S
- PullUp Enable para entradas
- Latch de Salida del Puerto
- Pin de Puerto

Programa ejemplo

```
      ldi   R16,0xFF      ;R16 = 0xFF = 0b11111111
      out DDRB,R16    ;pone Port B como salida(1111 1111)
L1: ldi  R16,0x55       ;R16 = 0x55 = 0b01010101
      out PORTB,R16   ;pone 0x55 en port B
      call  DELAY
      ldi   R16,0xAA      ;R16 = 0xAA = 0b10101010
      out   PORTB,R16   ;pone 0xAA en port B
      call DELAY
      rjmp L1
```

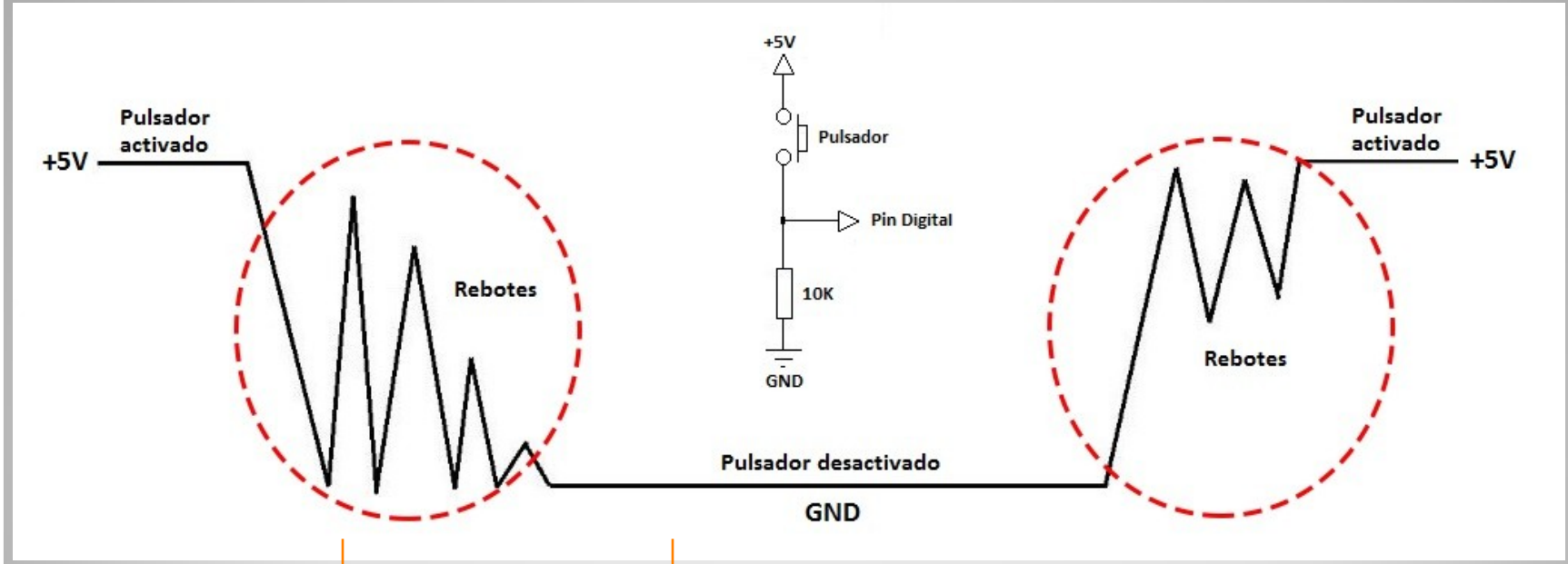
Aplicaciones básicas





Aplicaciones básicas

Rebote en pulsadores



Retardo en software (delay)
evita la captación de rebote

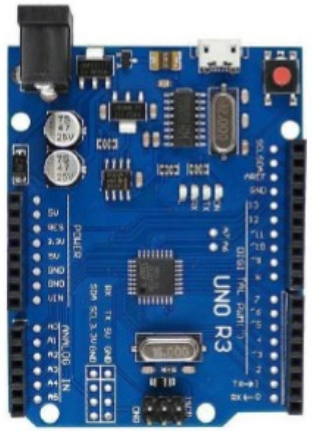
```
        cbi   DDRB,0      ;configura PB0 como entrada
        sbi   DDRB,5      ;configura PB5 como salida
AGAIN:  sbic PINB,0       ;salta la instrucción si PB0 es cero
        rjmp OVER
        cbi   PORTB,5
        rjmp AGAIN
OVER:   sbi   PORTB,5
        rjmp AGAIN
```

Ejemplo pin como entrada

- Se pueden usar los leds conectados a los pines
  13 (ptb5-enciente en alto),
  0 (ptd0- enciende en bajo)
- 1 (ptd1-enciende en bajo)

  El pulsador conectado al pin 11 (PB3)

Label	Port
A0	PC0(ADC0)
A1	PC1(ADC1)
A2	PC2(ADC2)
A3	PC3(ADC3)
A4	PC4(ADC4)
A5	PC5(ADC5)



Label	Port
SCL	PC5(ADC5/SCL)
SDA	PC4(ADC4/SDA)
AREF	AREF
GND	GND
13	PB5(SCK)
12	PB4(MISO)
11	PB3(MOSI)
10	PB2(OC1B)
9	PB1(OC1A)
8	PB0
7	PD7
6	PD6
5	PD5
4	PD4
3	PD3(INT1)
2	PD2(INT0)
1	PD1(TXD)
0	PD0(RXD)

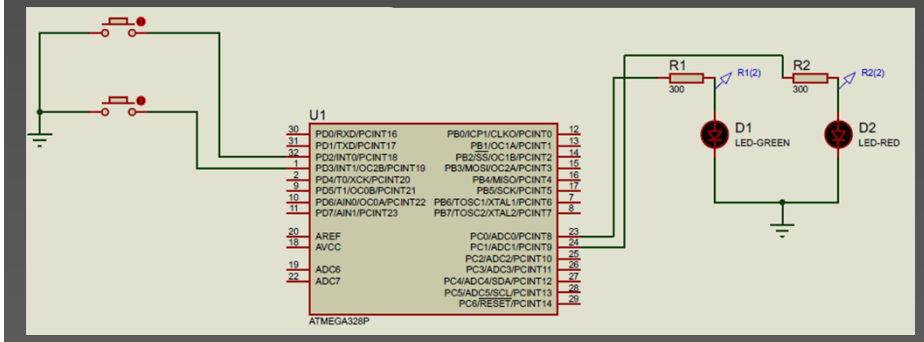
Leds y pulsadores en el kit

Interrupciones

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

```
Address     Labels Code                 Comments
0x0000             jmp    RESET         ; Reset Handler
0x0002             jmp    EXT_INT0      ; IRQ0 Handler
0x0004             jmp    EXT_INT1      ; IRQ1 Handler
0x0006             jmp    PCINT0        ; PCINT0 Handler
0x0008             jmp    PCINT1        ; PCINT1 Handler
0x000A             jmp    PCINT2        ; PCINT2 Handler
0x000C             jmp    WDT           ; Watchdog Timer Handler
0x000E             jmp    TIM2_COMPA    ; Timer2 Compare A Handler
0x0010             jmp    TIM2_COMPB    ; Timer2 Compare B Handler
0x0012             jmp    TIM2_OVF      ; Timer2 Overflow Handler
0x0014             jmp    TIM1_CAPT     ; Timer1 Capture Handler
0x0016             jmp    TIM1_COMPA    ; Timer1 Compare A Handler
0x0018             jmp    TIM1_COMPB    ; Timer1 Compare B Handler
0x001A             jmp    TIM1_OVF      ; Timer1 Overflow Handler
0x001C             jmp    TIM0_COMPA    ; Timer0 Compare A Handler
0x001E             jmp    TIM0_COMPB    ; Timer0 Compare B Handler
0x0020             jmp    TIM0_OVF      ; Timer0 Overflow Handler
0x0022             jmp    SPI_STC       ; SPI Transfer Complete Handler
0x0024             jmp    USART_RXC     ; USART, RX Complete Handler
0x0026             jmp    USART_UDRE    ; USART, UDR Empty Handler
0x0028             jmp    USART_TXC     ; USART, TX Complete Handler
0x002A             jmp    ADC           ; ADC Conversion Complete Handler
0x002C             jmp    EE_RDY        ; EEPROM Ready Handler
0x002E             jmp    ANA_COMP      ; Analog Comparator Handler
0x0030             jmp    TWI           ; 2-wire Serial Interface Handler
0x0032             jmp    SPM_RDY       ; Store Program Memory Ready Handler
;
0x0033     RESET:  ldi    r16, high(RAMEND); Main program start
0x0034             out    SPH,r16       ; Set Stack Pointer to top of RAM
0x0035             ldi    r16, low(RAMEND)
0x0036             out    SPL,r16
0x0037             sei                  ; Enable interrupts
0x0038     <instr> xxx
...    ...  ...    ...
```

Ejemplos Interrupciones



ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

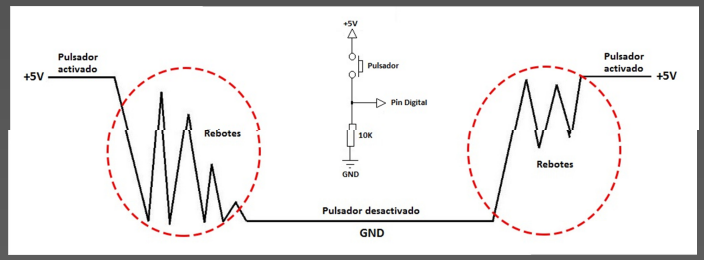
Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

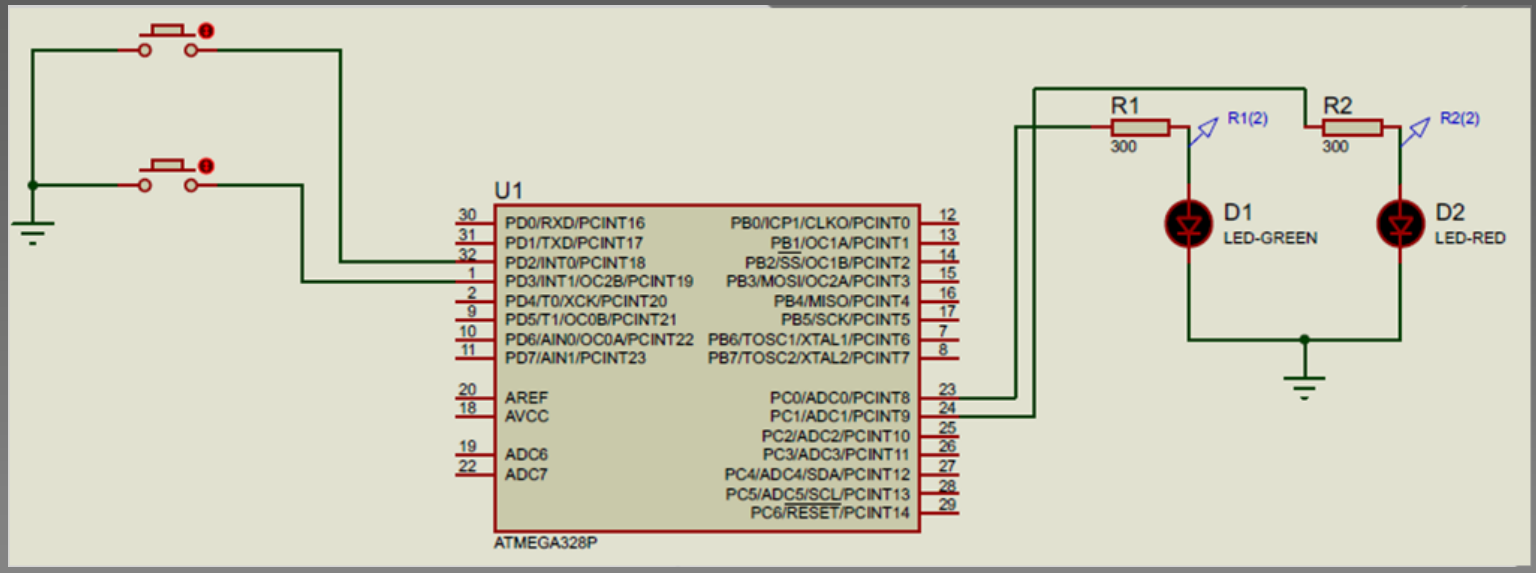
PCICR
PCMSK (0,1,2)
PCIFR
Para
demás puertos
PCINT:
(7..0) →Puerto B
(14..8) →Puerto C
(23..16) →Puerto D

```
/*
 * AssemblerApplication1.asm
 *
 *  Created: 28/09/2020 18:58:25
 *   Author:   Walter
 */
 rjmp Start
 rjmp Interr_0
 rjmp Interr_1

 Interr_0:
      in R14,$08      ; leo Puerto C
      com R14         ; complementa Puerto C
      out $08,R14
  reti

Interr_1:
      in R14,$08      ; leo Puerto C
      com R14         ; complementa Puerto C
      out $08,R14
   reti
```

```
Start:
      ldi R24,$F9
      out $0A,R24    ;  pines 2 y 3 de Puerto D como entrada

      ldi R24,$0C    ; PullUp enable
      out $0B,R24

      ldi R24,$03    ; Pines 0 y 1 de puerto C como salidas
      out $07,R24

      ldi R24,$00    ; Inicializo en 0 puerto C
      out $08,R24

      ldi R24,$00    ; EICRA en 0
      sts $69,R24

      ldi R24,$03    ; EIMSK en 3  habilita las interrup. Externas Int0 e Int1
      out $1D,R24

      ldi R24,$00    ; EIFR en 0   pone las banderas de interrupción en 0
      out $1C,R24

      sei

      Loop: rjmp Loop       ; bucle infinito
```

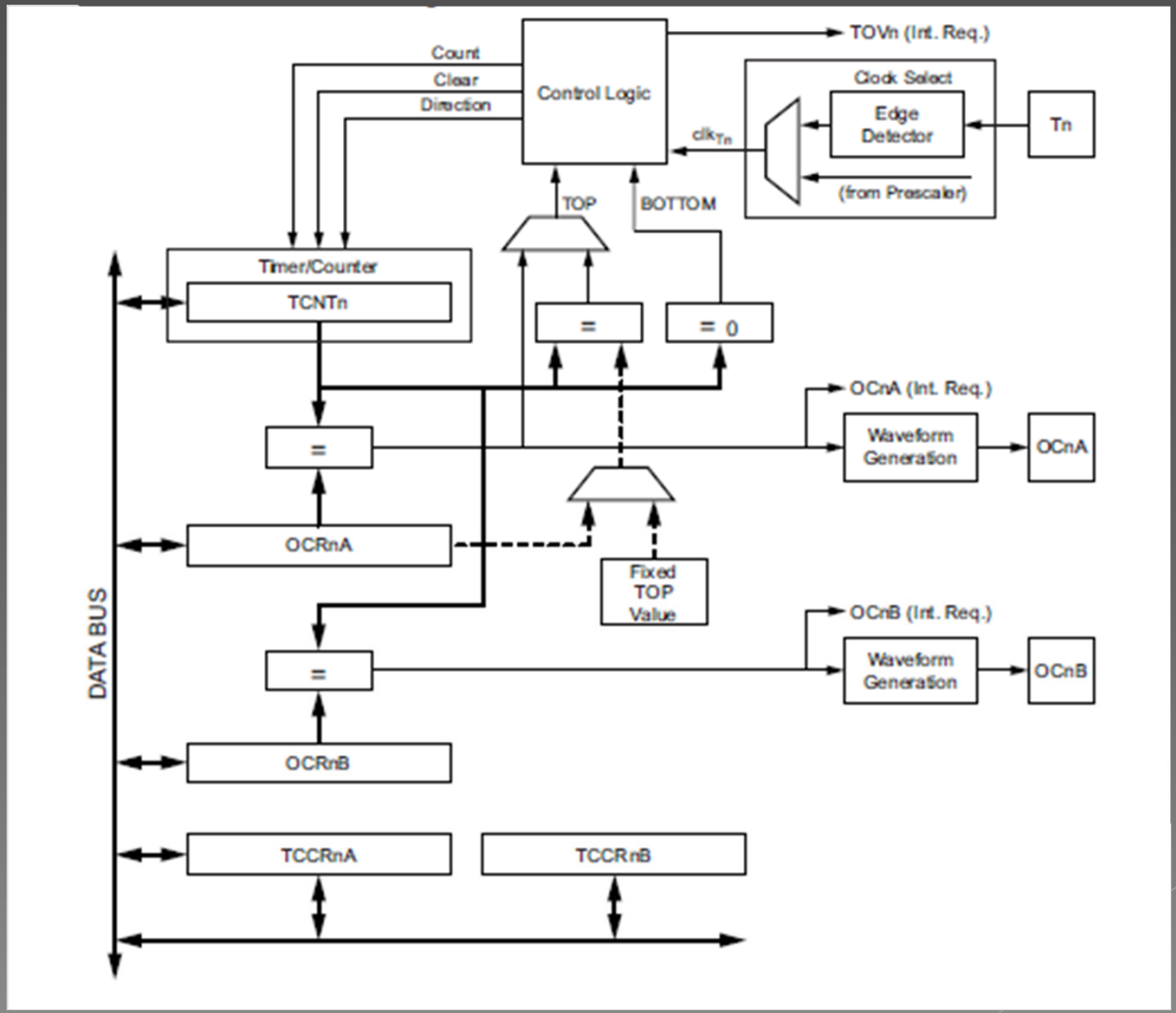
Ejemplos Interrupciones





Timers

Esquema general de los Timers



Timers

Banderas/Interrupciones

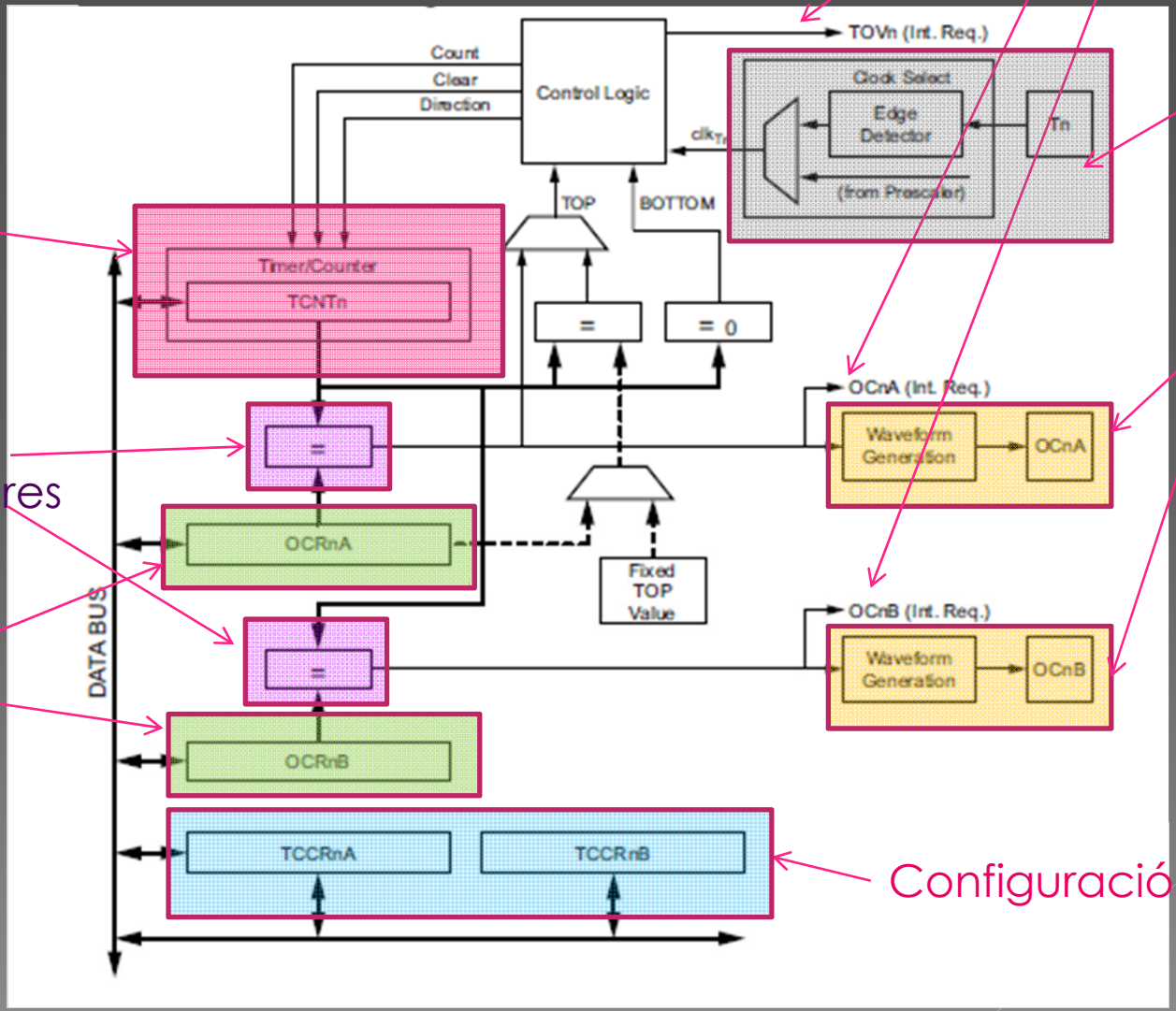
Fuente de Reloj y Prescaler

Contador (8 o 16 bits)

Salida a Pines

Comparadores

Registros a Comparar



Configuración

Timers

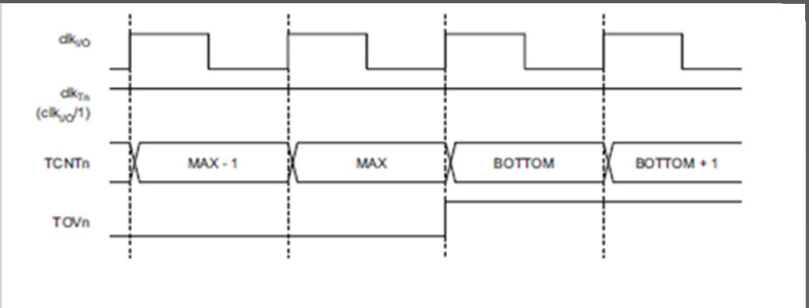
- Modos de Operación:

- Normal
- Limpiar Timer en Comparación (CTC)
- Modulación por Ancho de Pulso (PWM)
- Captura o Medición de Eventos

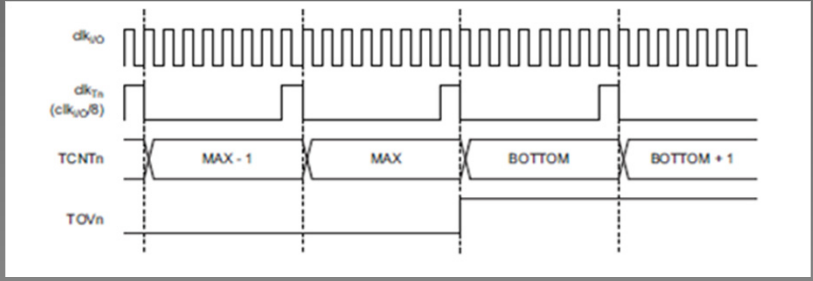
Timers

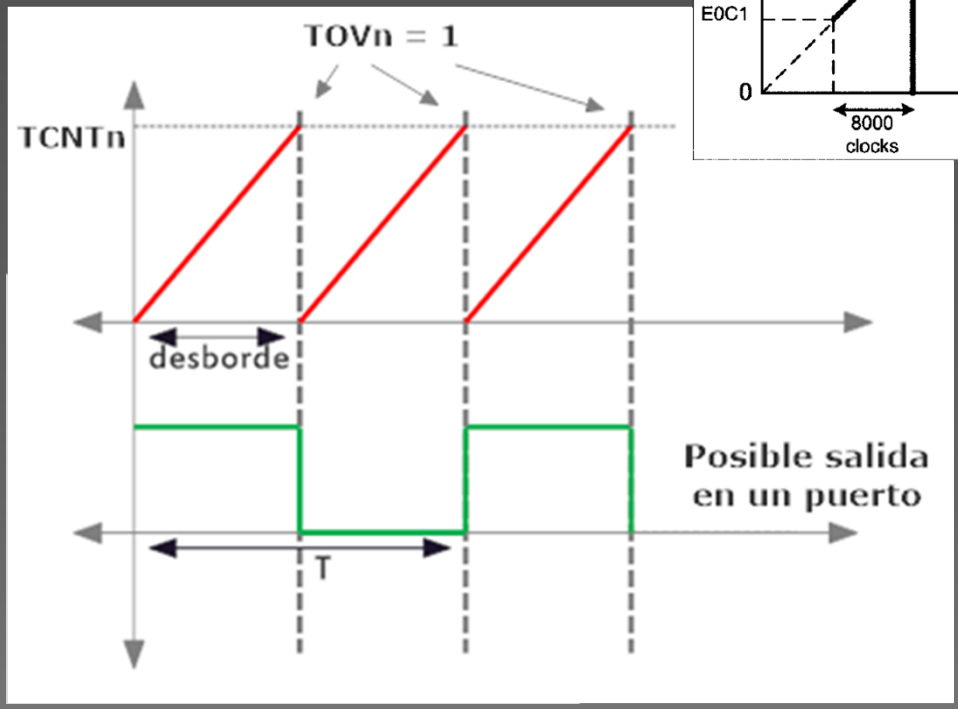
- Modo Normal

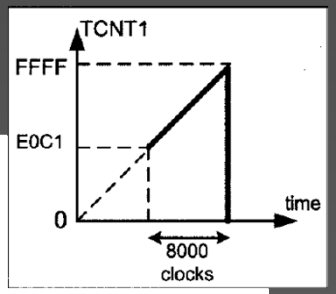
Sin Prescaler



Con Prescaler





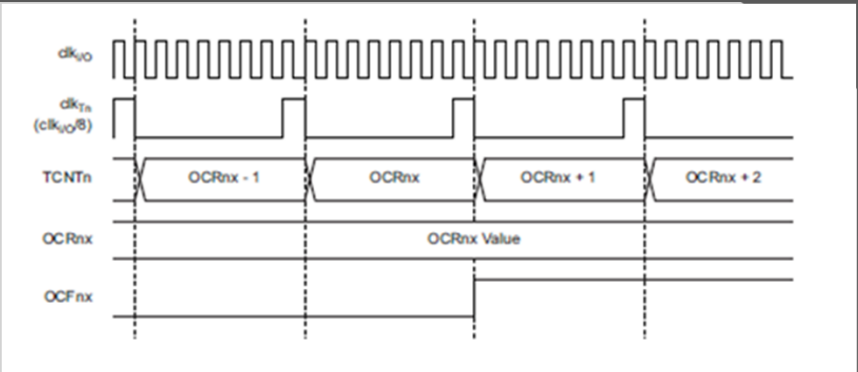


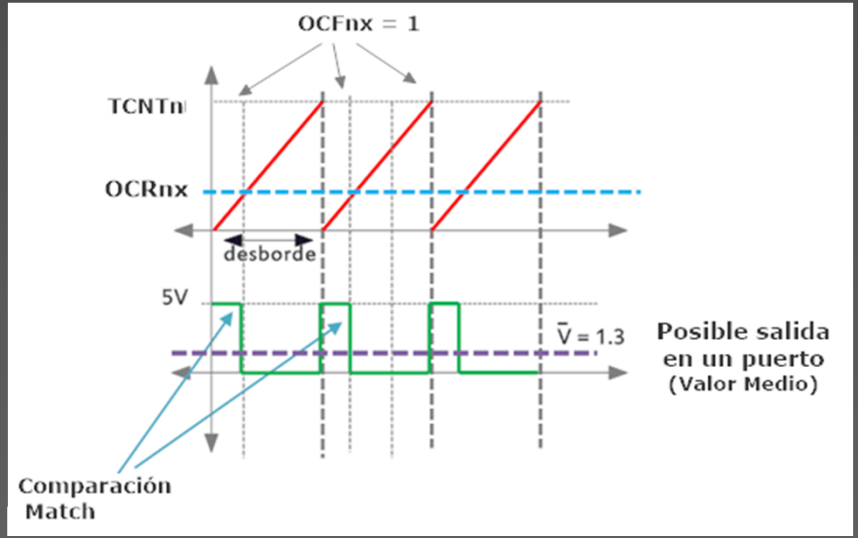
$T_{contador} = T_{reloj} * Prescaler$
(cada dígito del contador cambiará cada $T_{contador}$)

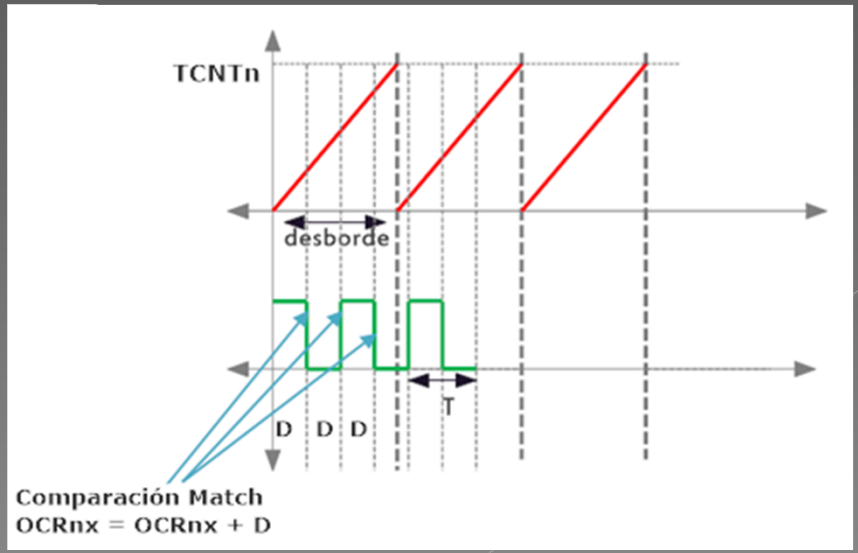
$Ts/2 = T_{contador} * (TCNT_{máx} + 1)$

Timers

- Modo Normal







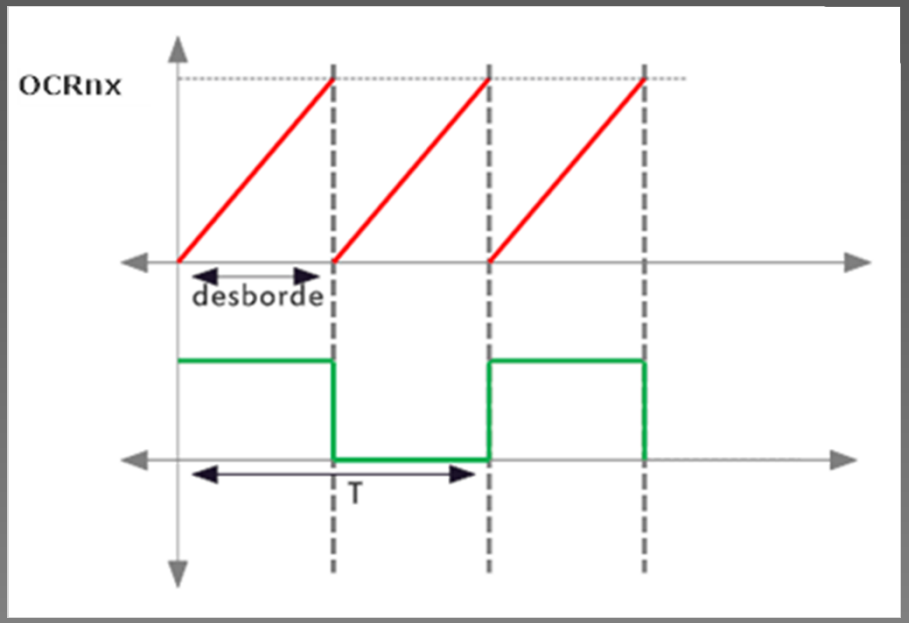
$T_{contador} = T_{reloj} * Prescaler$
(cada dígito del contador cambiará cada $T_{contador}$)

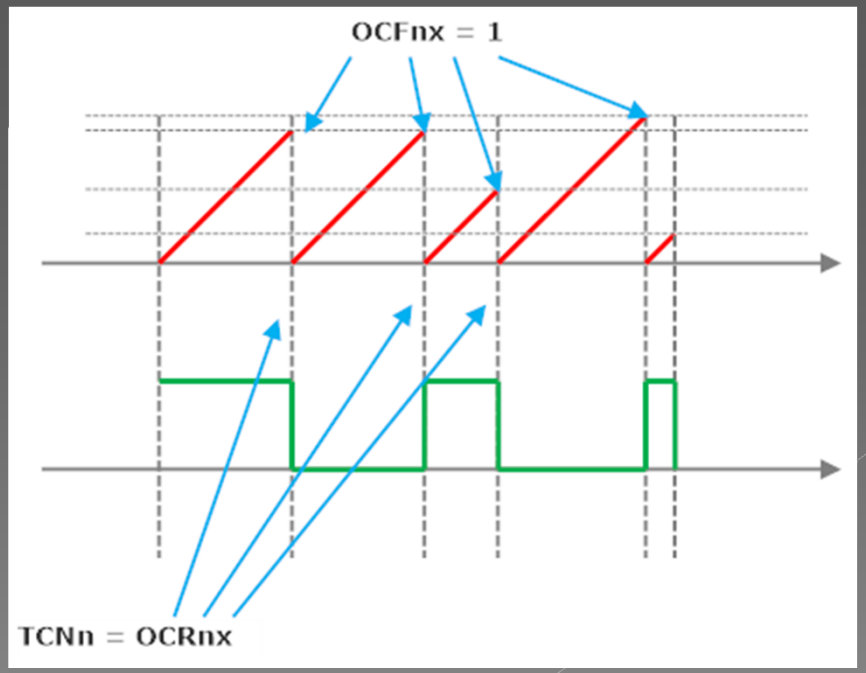
$Ts/2 = T_{contador} * (OCRnx + 1)$

Timers

- Modo CTC

El contador reinicia cuando su valor llega al de OCRnx

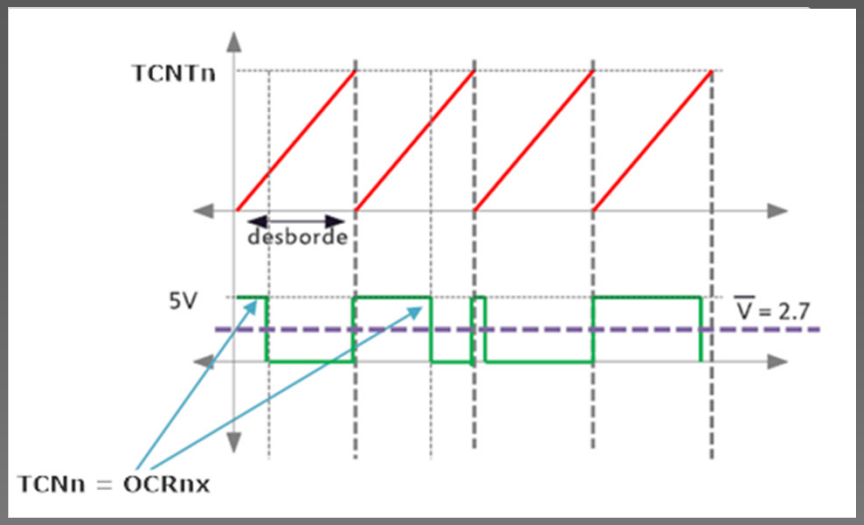




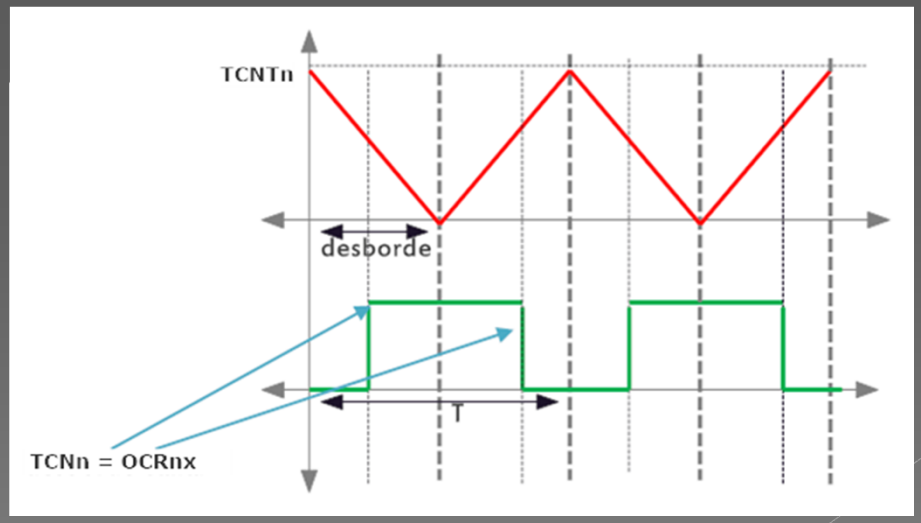
Timers

- Modo PWM

Fast PWM



PWM con corrección de fase



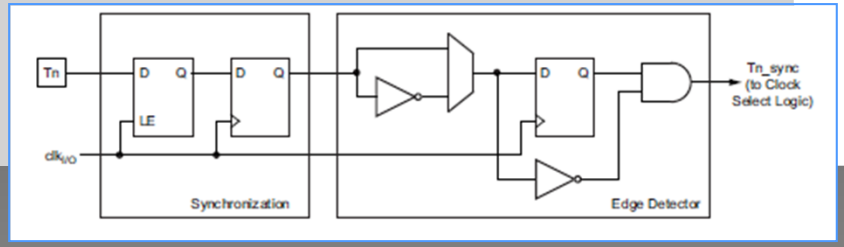
Timers

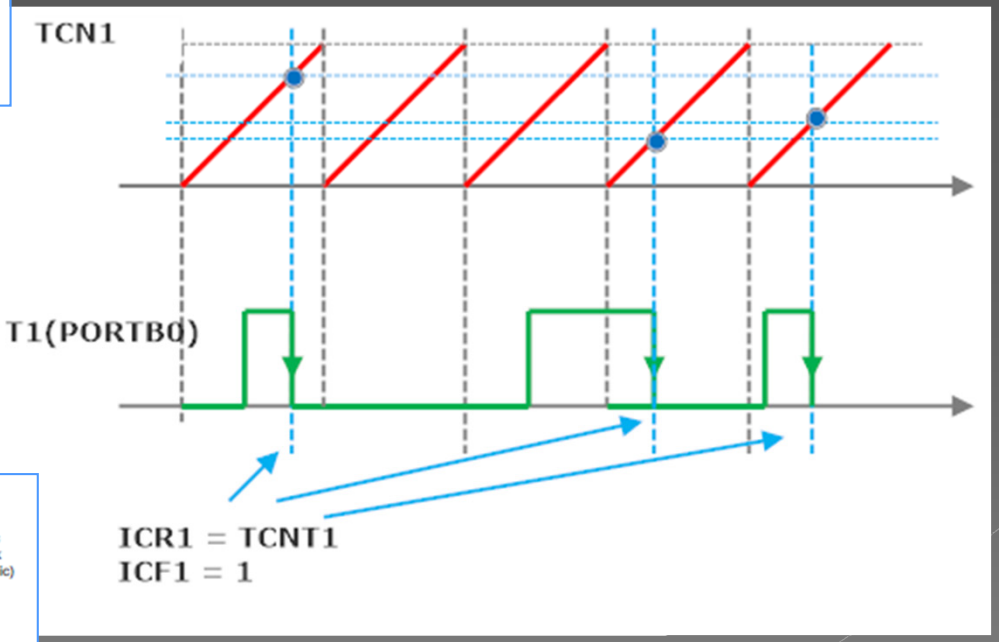
- Modo Captura de Eventos

Bit	7	6	5	4	3	2	1	0	
(0x87)	ICR1[15:8]								ICR1H
(0x86)	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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	





Timers

Configuración – Registros (Timer 0)

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	

Valor de Contador

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	

Registro de Configuración A

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	

Registro de Configuración B

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A[7:0]								OCR0A
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	

Valor a Comparar A

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	OCR0B[7:0]								OCR0B
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	

Valor a Comparar B

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

Habilitación de Interrupciones

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	

Banderas de Comparación

Timers

Configuración – Registros (Timer 0)

Registro de Configuración 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	

Registro de Configuración 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	

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

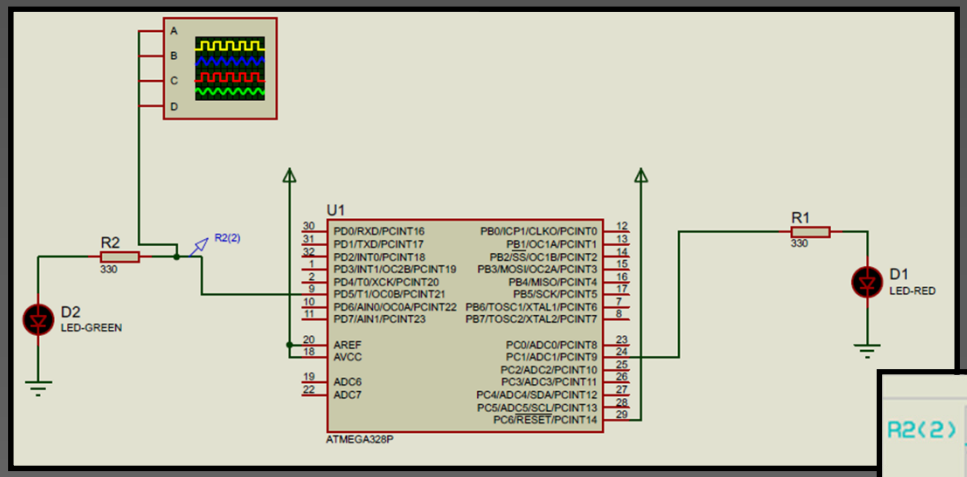
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

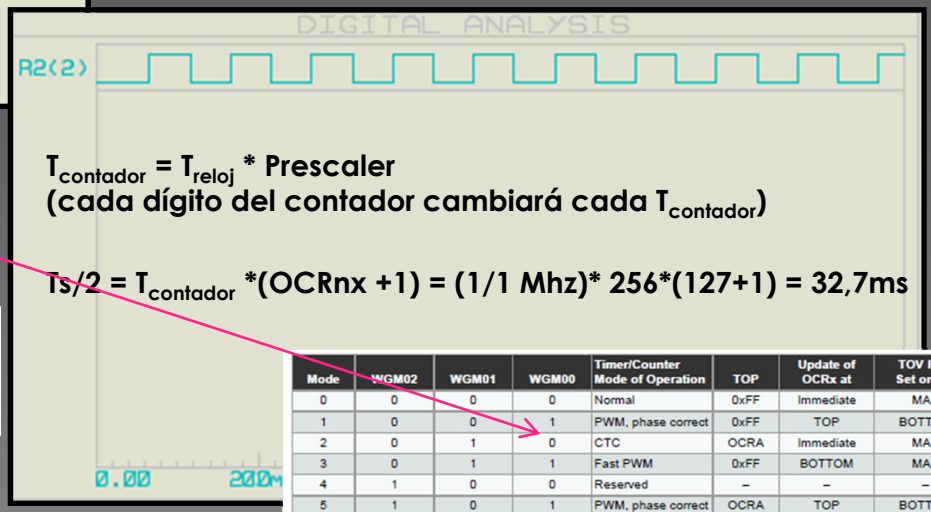
Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM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

Ejemplos Timers

Onda cuadrada simple sin interrupciones




```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

int main()
 {
  // Write your code here

   DDRD |= (1<<5) ;
   TCCR0A = 0b00010010 ;
   TCCR0B  = 0b00000100 ;

   OCR0B = 127 ;

  while(1)
   {;}

  return 0;
}
```



$T_{contador} = T_{reloj} * Prescaler$
(cada dígito del contador cambiará cada $T_{contador}$)

$Ts/2 = T_{contador} *(OCRnx +1) = (1/1\ Mhz)* 256*(127+1) = 32,7ms$

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	

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

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
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 on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

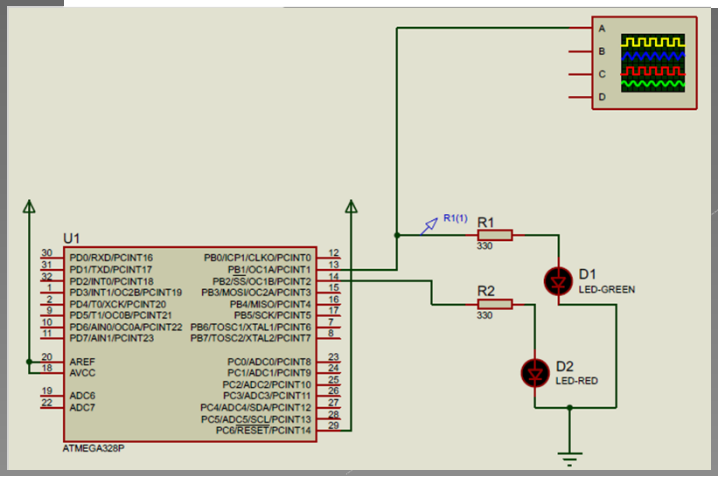
Ejemplos Timers

Destello simple con interrupciones en Timer 1 (16 bits)

```
;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================
    ; Reset Vector

     rjmp  Start

    .ORG 0X16
    rjmp interrupt_T1A

;=====================================================================
; CODE SEGMENT
;=====================================================================

Start:

 ; Write your code here

    ldi R20,$FF
    out $04,R20          ; dirección de puerto B como salidas
    ldi R20,$00
    out $05,R20          ; inicializa Puerto B en 0

    ldi R20,$40
    sts $80,R20          ; configura TCCR1A en CTC - prescaler 1024 reloj interno
    ldi R20,$0D          ;  TCCR1B
    sts $81,R20

    ldi R20,$00
   ;sts $85,R20          ; parte alta no usada
    ldi R20,$00
    sts $84,R20          ; contador en 0

     ldi R20,$01
     sts $89,R20         ; valor a conmparar OCR1AH
     ldi R20,$E7
     sts $88,R20         ; valor a conmparar OCR1AL
     ldi R20,$02         ; habrá una interrupción cada  500 ms a 1Mhz
     sts $6F,R20         ; habilita interrupt. OCIE1A
```

Así también puede hacerse un delay no bloqueante

```
        ldi   R20,0x0A     ; inicializa contador
        sei                ; habilita interrupciones generales

Loop:
        mov R19,R20        ;paso contador a otro registro
        breq prende        ;si el contador es 0 salta a prende
        cbi $05,2          ;pongo en 0 el bit 1 del puerto B
        rjmp Loop

prende:
        sbi $05, 2         pongo en 1 el bit 2 del puerto B
        ldi   R20,0x0A     vuelvo a cargar el contador en 10
        rjmp  Loop
;================================================
interrupt_T1A:
            dec R20    ; decremento el contador
            reti


; El LED rojo destellará cada 10 interrupciones.
```

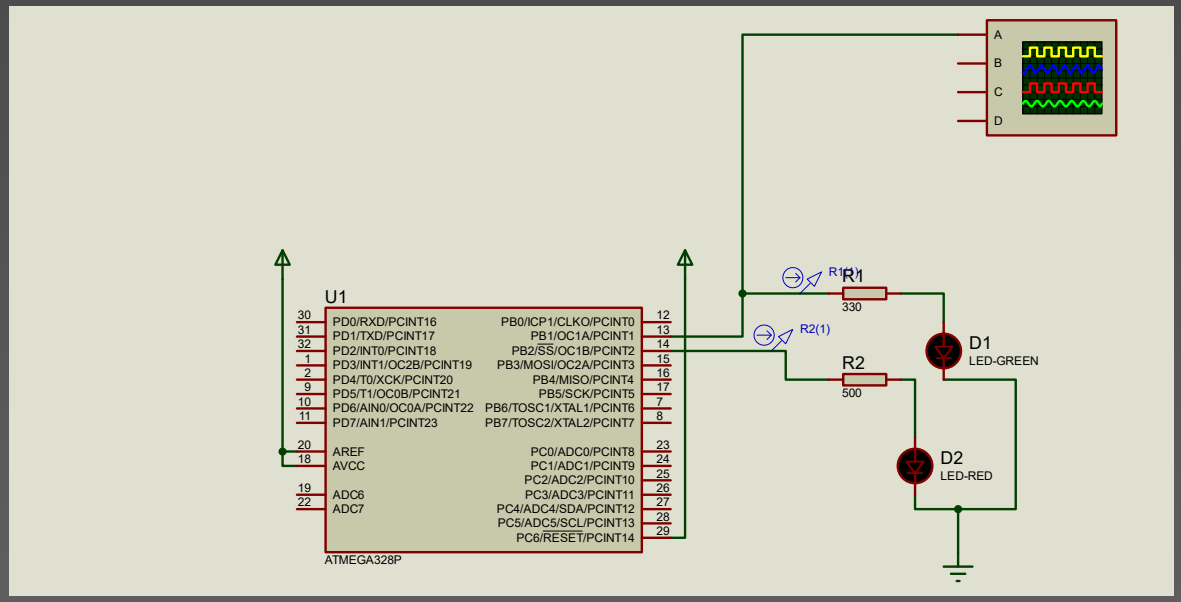


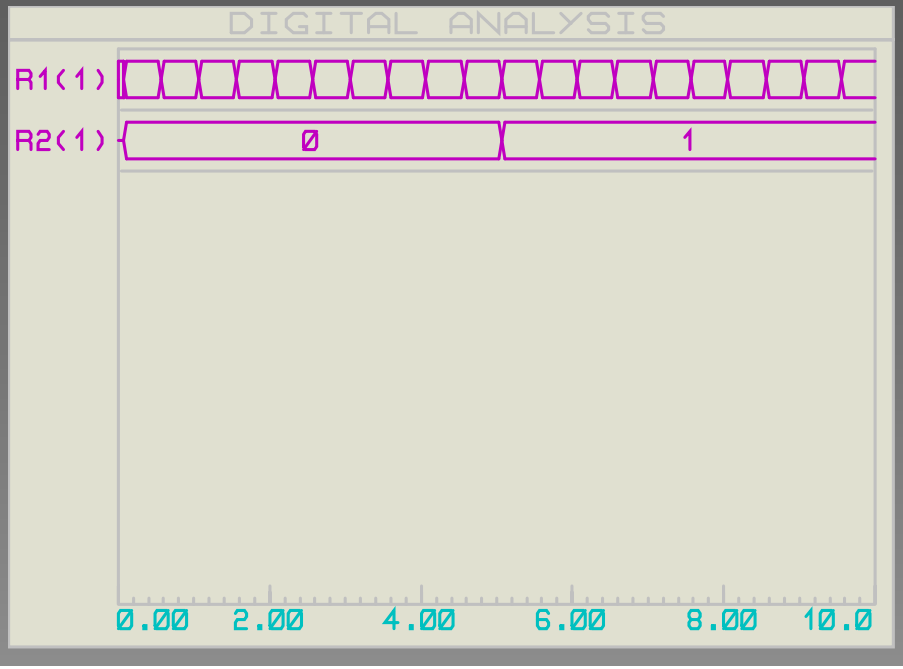
Señal cuadrada simple con interrupciones en Timer 1 (16 bits)

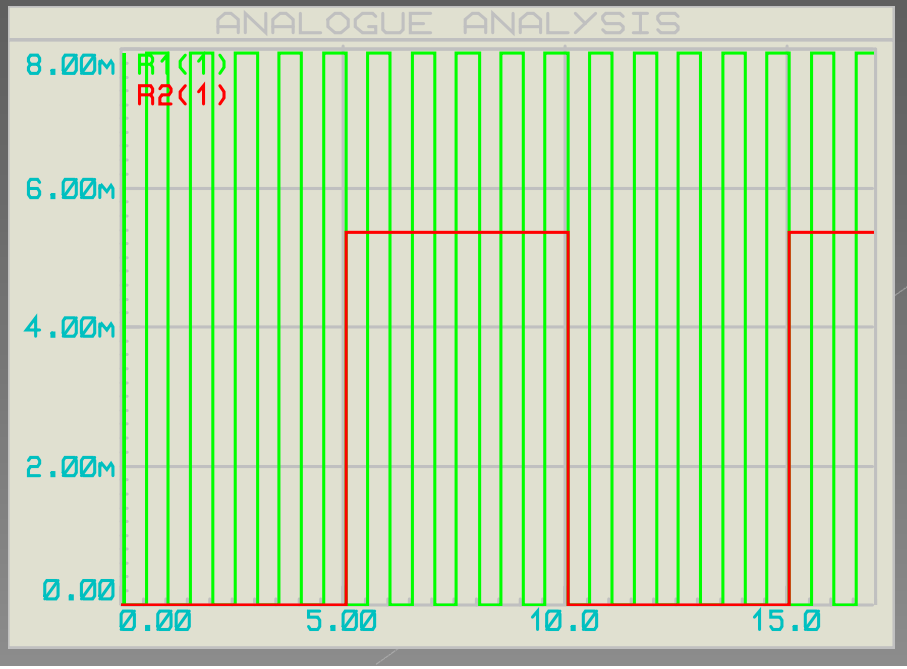
```
;
; Assembler_Timer_formateado.asm
;
; Created: 24/10/2020 10:58:40
; Author : Walter
;

.include "M328PDEF.INC"

;====================================================
; RESET and INTERRUPT VECTORS
;====================================================

     ; Reset Vector
      rjmp  Start

     .ORG 0X16      ; posición del vector de Timer que usaremos
     rjmp interrupt_T1A


; Replace with your application code

;====================================================
; CODE SEGMENT
;====================================================

start:

;====================================================
; Inicializo Pila

   ldi R16,HIGH(RAMEND)
   out SPH,R16
   ldi R16,LOW(RAMEND)
   out SPL,R16

  ;====================================================
 ;configuración de puertos y Timer
  ;====================================================
     ldi R20,$06
     out DDRB,R20            ; dirección de puerto B como salidas
     ldi R20,$00
     out PORTB,R20           ; inicializa Puerto B en 0

     ldi R20,$40
     sts TCCR1A,R20          ; configura TCCR1A y TCCR1B en CTC
                             ; prescaler  1024 reloj interno
     ldi R20,0b00001101      ; $0D
     sts TCCR1B,R20
```

```
     ldi R20,$00             ; inicializo contador en 0 (opcional)
     sts TCNT1H,R20          ; parte alta
     ldi R20,$00
     sts TCNT1L,R20          ; parte baja contador en 0

     ldi R20,$01
     sts OCR1AH,R20          ; valor a conmparar OCR1AH
     ldi R20,$E7             ; habrá una interrupción cada  500 ms
     sts OCR1AL,R20          ; valor a conmparar OCR1AL

     ldi R20,$02             ;
     sts TIMSK1,R20          ; habilita interrupt. OCIE1A
     ldi R20,$0A             ;inicializa contador
     sei                     ; habilita interrupciones generales

Loop:
     mov R19,R20             ;paso contador a otro registro
     breq prende             ;si el contador es 0 salta a prende
     ;cbi PORTB,2            ;pongo en 0 el bit 2 del puerto B
   rjmp Loop

prende:
     ldi R17,$04             ;máscara para invertir bit 2 de PB
     in  R16,PINB            ; leo puerto B
     eor R16,R17             ; invierto el bit 2
     out PORTB,R16           ; almaceno valor en puerto
     ;sbi PORTB, 2           ;pongo en 1 el bit 2 del puerto B

     ldi R20,0x0A            ;vuelvo a cargar el contador en 10
   rjmp Loop

     rjmp start

;====================================================
; Rutina de Atención de Interrupción
;====================================================

interrupt_T1A:
            dec R20     ; decremento el contador
            sbi PINB,1
             reti
; El LED verde conmutará cada 0,5 segundos  (clk = 1Mhz)
; El LED rojo destellará cada 10 interrupciones.
```

Señal cuadrada simple con interrupciones en Timer 1 (16 bits)







Destello simple con interrupciones en Timer 1 (16 bits) - Macros

```
; Timer_macros.asm
;
; Created: 24/10/2020 14:40:42
; Author : Walter
;
 .include "M328PDEF.INC"

.macro cargaR
   ldi R20,@1
   out @0,R20
.endmacro

.macro cargaM
   ldi R20,@1
   sts @0,R20
.endmacro

.macro setPila
   ldi R16,HIGH(RAMEND)
   out SPH,R16
   ldi R16,LOW(RAMEND)
   out SPL,R16
.endmacro

;=================================================
; RESET and INTERRUPT VECTORS
;=================================================

      ; Reset Vector
       rjmp  Start

.ORG 0X16       ; posición del vector de Timer que usaremos
      rjmp interrupt_T1A


; Replace with your application code

;=================================================
; CODE SEGMENT
;=================================================

start:

;=================================================
; Inicializo Pila

  setPila
```

.include "M328PDEF.INC"
.include "misMacros.inc"

```
;=================================================
    ;configuración de puertos y Timer
 ;=================================================

  cargaR DDRB,$FF          ; dirección de puerto B como salidas
  cargaR PORTB,$00         ; inicializa Puerto B en 0

  cargaM TCCR1A,$40        ; configura TCCR1A y TCCR1B
  cargaM TCCR1B,$0D        ; $0D

     ; inicializo contador en 0 (opcional)
   cargaM TCNT1H,$00       ; parte alta
   cargaM TCNT1L,$00        ;  parte baja contador en 0
                            ; habrá una interrupción cada  500 ms
   cargaM OCR1AH,$00        ; valor a conmparar OCR1AL
   cargaM OCR1AL,250        ; valor a conmparar OCR1AH
   cargaM TIMSK1,$02        ; habilita interrupt. OCIE1A

  sei                       ; habilita interrupciones generales

Loop:
      mov R19,R20           ;paso contador a otro registro
      breq prende           ;si el contador es 0 salta a prende
      cbi PORTB,5           ;pongo en 0 el bit 5 del puerto B
   rjmp Loop

 prende:
       sbi PORTB,5          ;pongo en 1 el bit 5 del puerto B
       ldi R20,0x0A         ;vuelvo a cargar el contador en A
      rjmp Loop

      rjmp start
;=================================================
; Rutina de Atención de Interrupción
;=================================================

interrupt_T1A:
            dec R20     ; decremento el contador
            sbi PINB,1
            reti
; El LED verde conmutará cada 0,5 segundos  (clk = 1Mhz)
; El LED rojo destellará cada 10 interrupciones.
```

Modo Sleep: Registro

- **Se implementará Modo Sleep en Power Down (SM1=1 y SE=1)**

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC noise reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	External standby(1)

Note: 1. Standby mode is only recommended for use with external crystals or resonators.

Modo Sleep: Ejemplo

```
.ORG 000              ;vector de reset
   JMP MAIN
.ORG INT0addr         ;registro de INT0
   JMP EX0_ISR
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20 ;initializa stack
```

STACK

```
  LDI R20,0x2 ;configura INT0 activo en bajo
  STS EICRA,R20
  SBI PORTD,2 ;pull-up activado
  LDI R20,1<<INT0 ;habilita INT0
  OUT EIMSK,R20
```

Configuración de interrupción externa INT0

```
  SBI DDRB,5 ;PORTB.5 = salida
ldi r16,0x5
  out SMCR, r16
```

Modo Sleep - Power Down (SM1=1 y SE=1)

```
  SEI ;habilita interrupciones

HERE:
  sleep ; entra en sleep y espera una interrupción ;
  JMP HERE
```

Bucle Infinito

Simulación

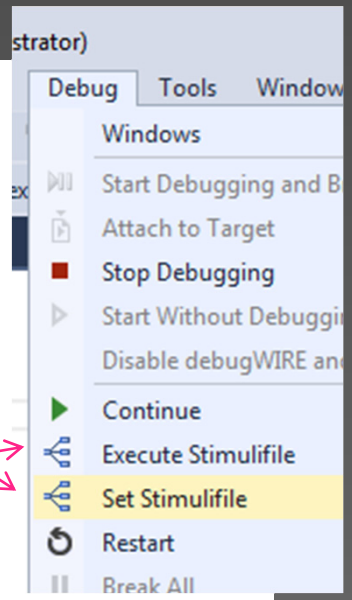
- SE simulará un evento externo en el pin de INT0 mediante un archivo stimulifile
- Crear un archivo de texto y agregar las siguientes líneas:

```
// Setea el PIND2 (INT0) en alto y bajo cada 20 ciclos
$repeat 2000000
   PIND |= 0x04
   #20
   PIND = 0x0
   #20
$endrep
```

- Poner extensión .stim

Debugging

- Entrar al Debug
- Seleccione “Set Stimulifile”
- Abra el archivo .stim
- Seleccione “Execute Stimulifile”

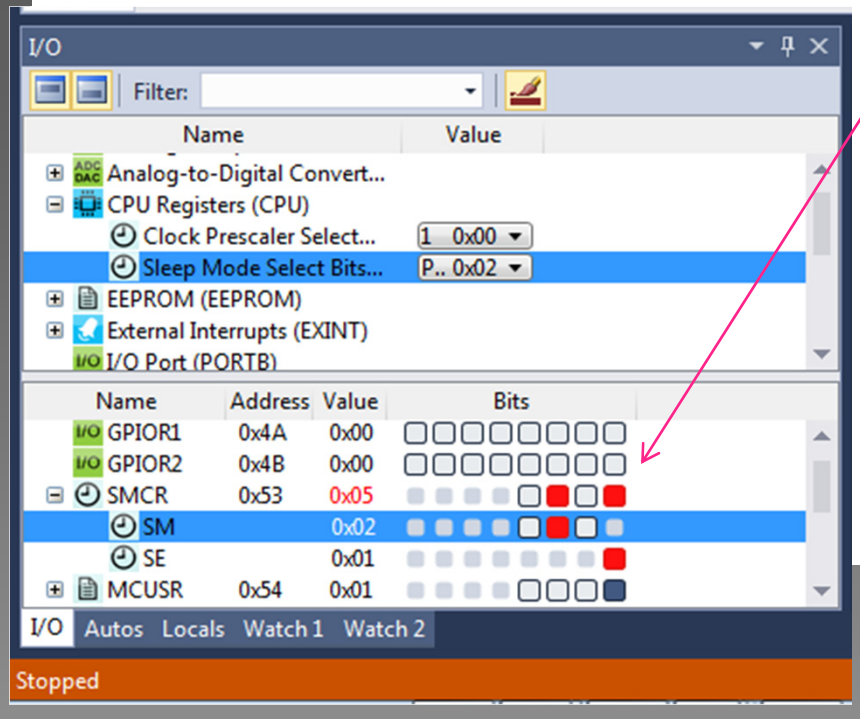


Uso de Breakpoints para interrupciones

- Para visualizar el acceso a la interrupción se debe poner un breakpoint dentro de la rutina
- Se puede ejecutar Paso a paso hasta

Sentencia sleep para ver el efecto de las configuraciones en la ventana I/O

```
main.asm  pulsador_led_int
    SBI PORTD,2 ;pull-up activado
    LDI R20,1<<INT0 ;habilita INT0
    OUT EIMSK,R20
    ldi r16,0x5
    out SMCR, r16
    SEI ;habilita interrupciones
HERE:
    sleep ; entra en sleep, espera una interrupción ;
    JMP HERE


EX0_ISR:
    CLI
    in r21,portb
    ldi r22,(1<<5) ;para cambiar estado
    eor r21,r22
    out portb,r21
    SEI
    RETI
```



Uso de Breakpoints para interrupciones

- Presionar en Continue (F5)
- Se ejecutará la simulación en tiempo real y se detendrá en el breakpoint dentro de la rutina interrupción
- A partir de ahí se puede ejecutar paso a paso para comprobar la ejecución del código de la rutina