

## **TEMPORIZADOR / CONTADORES**

### **Descripción general del temporizador AVR® MCU**

Los temporizadores son una característica muy útil de un microcontrolador para contar pulsos en un pin de entrada. Cuando son impulsados por el reloj de instrucciones, pueden convertirse en una base de tiempo precisa. Los dispositivos AVR® tienen temporizadores de 8 bits y 16 bits de ancho y ofrecen diferentes características basadas en el dispositivo. Un conjunto muy típico de temporizadores se puede encontrar en el microcontrolador AVR **ATmega328PB**. Este dispositivo tiene cinco temporizadores / contadores como se describe aquí:

Timer 0	TC0	8-bit Timer/counter with Pulse Width Modulation (PWM)
Timer 1	TC1	16-bit Timer/counter with PWM and Asynchronous Operation
Timer 2	TC2	8-bit Timer/counter with PWM and Asynchronous Operation
Timer 3	TC3	16-bit Timer/counter with PWM and Asynchronous Operation
Timer 4	TC4	16-bit Timer/counter with PWM and Asynchronous Operation

### **Definiciones:**

#### **Nomenclatura de registros**

**FONDO** El contador llega al BOTTOM cuando se convierte en cero (0x00 para contadores de 8 bits y 0x0000 para contadores de 16 bits)

**Máximo** El contador alcanza su valor máximo cuando se convierte en 0x0F (15 decimales) para contadores de 8 bits y 0x00FF (255 decimales) para contadores de 16 bits

**Arriba** El contador llega al TOP cuando su valor se vuelve igual al valor más alto posible. Al valor TOP se le puede asignar un valor fijo MAX o el valor almacenado en el registro OCRxA. Esta asignación depende del modo de funcionamiento

### **TC0 - Temporizador/Contador de 8 bits con PWM**

Timer/Counter0 (TC0) es un módulo de temporizador/contador de 8 bits de uso general, con dos unidades de comparación de salida independientes y soporte PWM.

### **Registros TC0**

El registro Timer/Counter 0 (TCNT0) y los registros Output Compare TC0x (OCR0x) son registros de 8 bits. Las señales de solicitud de interrupción son todas visibles en el Registro de indicador de interrupción del temporizador 0 (TIFR0). Todas las interrupciones se enmascaran individualmente con el Registro de máscara de interrupción del temporizador 0 (TIMSK0).

**Name:** TCCR0B

**Offset:** 0x45

**Reset:** 0x00

**Property:** When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS02	CS01	CS00
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

**Fuentes de temporizador/contador de reloj TC0**

TC0 puede ser sincronizado por una fuente de reloj interna o externa. La fuente de reloj se selecciona escribiendo en los bits de selección de reloj (CS02:0) en el registro de temporizador/contador de control (TCCR0B).

**Bits 2:0 – CS0n: Selección de reloj [n = 0..2]**

Los tres bits de selección de reloj seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Description
0	0	0	No Clock source (Timer stopped)
0	0	1	clkio/1 (No prescaling)
0	1	0	clkio/8 (From prescaler)
0	1	1	clkio/64 (From prescaler)
1	0	0	clkio/256 (From prescaler)

1	0	1	clkio/1012 (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)

### Unidad de contador TC0

Dependiendo del modo de operación utilizado, T0 se borra, incrementa o disminuye en cada reloj temporizador (clkT0). clkT0 se puede generar a partir de una fuente de reloj externa o interna, seleccionada por los bits clock Select (CS0[2:0]).

La secuencia de conteo está determinada por la configuración de los bits WGM01 y WGM00 ubicados en el Registro de Control T0 A (TCCR0A) y el bit WGM02 ubicado en el Registro de Control de Temporizador/Contador B (TCCR0B).

### Bits 1:0 – WGM0n: Modo de generación de forma de onda [n = 1:0]

Combinados con el bit WGM02 que se encuentra en el registro TCCR0B, estos bits controlan la secuencia de conteo del contador, la fuente del valor máximo del contador (TOP) y qué tipo de generación de forma de onda se utilizará. Los modos de operación admitidos por la unidad de temporizador / contador son: modo normal (contador), modo de tiempo de borrado en modo de coincidencia de comparación (CTC) y dos tipos de modos de modulación de ancho de pulso (PWM).

Tabla 1-2 Descripción del bit del modo de generación de forma de onda

Mode	WGM2:0	Mode of Operation	TOP	OCR0x Update	TOV flag set on
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	MAX

Note:

1. MAX = 0xFF
2. BOTTOM = 0x00

### Modos de funcionamiento para TC0

El modo de operación determina el comportamiento de tc0 y los pines de comparación de salida. Se define por la combinación de los bits de modo de generación de forma de onda y los bits de modo de comparación de salida en los registros de control de temporizador/contador A y B (TCCR0B.WGMn2, TCCR0A.WGM01, TCCR0A.WGM00 y TCCR0A.COM0x[1:0]).

Los modos de operación disponibles para TC0 son:

1. Modo normal
2. Borrar temporizador en el modo Comparar coincidencia (CTC)
3. Modo PWM rápido
4. Modo PWM de fase correcta

### Borrar temporizador en el modo Comparar coincidencia

En el modo Clear Timer on Compare o CTC (WGM0[2:0]=0x2), el **registro OCR0A** se utiliza para manipular la resolución del contador: el contador se borra a CERO cuando el valor del contador (TCNT0) coincide con el OCR0A. El OCR0A define el valor superior para el contador, de ahí también su resolución.

El valor del contador (TCNT0) aumenta hasta que se produce una coincidencia de comparación entre TCNT0 y OCR0A y luego se borra el contador (TCNT0). Se puede generar una interrupción cada vez que el valor del contador alcanza el valor TOP estableciendo el indicador OCF0A. Si la interrupción está habilitada, la rutina del controlador de interrupciones se puede utilizar para actualizar el valor TOP.

La frecuencia de la forma de onda se define mediante la siguiente ecuación:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

N representa el factor preescalador (1, 8, 64, 256 o 1024).

### TC1, TC3 y TC4 - Temporizador/Contadores de 16 bits con PWM

Las unidades timer/counter de 16 bits permiten un tiempo preciso de ejecución del programa (gestión de eventos), generación de ondas y medición del tiempo de señal.

#### Registros (TC1, TC3, TC4)

1. El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son registros de 16 bits.
2. Los registros de control de temporizador/contador (TCCRnA/B) son registros de 8 bits y no tienen restricciones de acceso a la CPU.
3. Las señales de solicitudes de interrupción (abreviadas como Int.Req. en el diagrama de bloques) son todas visibles en el Registro de indicadores de interrupción del temporizador (TIFRn). Todas las interrupciones se enmascaran individualmente con el Registro de máscaras de interrupción del temporizador (TIMSKn).

#### Fuentes de temporizador/contador de reloj (TC1, TC3, TC4)

El temporizador/contador puede ser sincronizado por una fuente de reloj interna o externa. La fuente del reloj se selecciona mediante la lógica Clock Select que está controlada por los bits Clock Select en el temporizador/Contador de control Registro B (TCCRnB.CS[2:0]).

Name: TCCR1B, TCCR3B, TCCR4B							
Offset: 0x81 + n*0x10 [n=0..2]							
Reset: 0x00							
Property: -							
Bit	7	6	5	4	3	2	1 0
	ICNC	ICES		WGM3	WGM2	CS[2:0]	
Access	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0

Bits 2:0 – CS[2:0]: Clock Select [n = 0..2] Los tres bits de selección de reloj seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Description
0	0	0	No Clock source (Timer stopped)
0	0	1	clkio/1 (No prescaling)

0	1	0	clkio/8 (From prescaler)
0	1	1	clkio/64 (From prescaler)
1	0	0	clkio/256 (From prescaler)
1	0	1	clkio/1012 (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)

### Unidad de contador (TC1, TC3, TC4)

TC1, TC3 y TC4 son contadores bidireccionales programables de 16 bits.

Cada contador de 16 bits se asigna en dos ubicaciones de memoria de E/S de 8 bits: **Counter High** (TCNTnH) que contiene los ocho bits superiores del contador y **Counter Low** (TCNTnL) que contiene los ocho bits inferiores.

Dependiendo del modo de operación seleccionado, el contador se borra, incrementa o disminuye en cada reloj temporizador (clkTn). El clkTn de reloj se puede generar a partir de una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj en el registro de control de temporizador/contador B (TCCRnB.CS[2:0]).

La secuencia de conteo está determinada por la configuración de los bits del modo de generación de forma de onda en los registros de temporizador/contador de control A y B (TCCRnB.WGM[3:2] y TCCRnA.WGM[1:0]).

### Modos de funcionamiento (TC1, TC3, TC4)

El modo de operación está determinado por la combinación de los bits de modo de generación de forma de onda (WGM[3:0]) y modo de comparación de salida (TCCRnA.COMx[1:0]).

Los modos de operación disponibles son:

1. Modo normal
2. Borrar temporizador en el modo Comparar coincidencia (CTC)
3. Modo PWM rápido
4. Modo PWM de fase correcta
5. Modo PWM correcto de fase y frecuencia

## Modo PWM rápido

Los modos Fast Pulse Width Modulation o Fast PWM (modos 5, 6, 7, 14 y 15, WGM[3:0]= 0x5, 0x6, 0x7, 0xE, 0xF) proporcionan una opción de generación de forma de onda PWM de alta frecuencia. El Fast PWM se diferencia de las otras opciones de PWM por su operación de una sola pendiente. El contador cuenta de BOTTOM a TOP y luego se reinicia desde BOTTOM.

En el modo PWM rápido, el contador se incrementa hasta que el valor del contador coincide con uno de los valores fijos 0x00FF, 0x01FF o 0x03FF (WGM[3:0] = 0x5, 0x6 o 0x7), el valor en ICRn (WGM[3:0]=0xE) o el valor en OCRnA (WGM[3:0]=0xF). El contador se borra en el siguiente ciclo de reloj del temporizador.

La frecuencia PWM para la salida se puede calcular mediante la siguiente ecuación:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

Nota:• La "n" en los nombres de registro y bit indica el número de dispositivo (n = 0 para temporizador/contador 0) y la "x" indica unidad de comparación de salida (A/B).• N representa el divisor de preescala (1, 8, 64, 256 o 1024).

## TC2 - Temporizador/Contador2 de 8 bits con PWM y operación asincrónica

Timer/Counter2 (TC2) es un módulo de timer/counter de 8 bits de doble canal y uso general.

### Registros TC2

El temporizador/contador (TCNT2) y el registro de comparación de salida (OCR2A y OCR2B) son registros de 8 bits. Las señales de solicitud de interrupción son todas visibles en el Registro de indicadores de interrupción del temporizador (TIFR2). Todas las interrupciones se enmascaran individualmente con el Registro de máscaras de interrupción del temporizador (TIMSK2).

Name: TCCR2B

Offset: 0xB1

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	FOC2A	FOC2B			WGM22	CS22	CS21	CS20
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

## Fuentes de reloj TC2

TC2 puede ser sincronizado por una fuente de reloj síncrona interna o asincrónica externa: Los tres bits de selección de reloj (CS2: CS0) seleccionan la fuente de reloj que utilizará el temporizador / contador.

CS02	CS01	CS00	Description
0	0	0	No Clock source (Timer stopped)
0	0	1	clkio/1 (No prescaling)
0	1	0	clkio/8 (From prescaler)
0	1	1	clkio/64 (From prescaler)
1	0	0	clkio/256 (From prescaler)
1	0	1	clkio/1012 (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)

### Unidad de contador TC2

Dependiendo del modo de operación utilizado, el contador se borra, incrementa o disminuye en cada reloj temporizador (clkT2). clkT2 se puede generar a partir de una fuente de reloj externa o interna, seleccionada por los bits clock Select (CS2[2:0]). La secuencia de conteo está determinada por la configuración de los bits WGM21 y WGM20 ubicados en el Registro de Control de Temporizador/Contador (TCCR2A) y el bit WGM22 ubicado en el Registro de Control de Temporizador/Contador B (TCCR2B).

### Modos de funcionamiento de TC2

El modo de operación, es decir, el comportamiento del temporizador/contador y los pines de comparación de salida, se define mediante la combinación de los bits de modo de generación de



forma de onda (WGM2[2:0]) y modo de comparación de salida (COM2x[1:0]). Los modos de operación disponibles son:

1. Modo normal
2. Borrar temporizador en el modo Comparar coincidencia (CTC)
3. Modo PWM rápido
4. Modo PWM de fase correcta

### **Modo normal**

En el modo Normal (WGM22:0 = 0) la dirección de conteo siempre está arriba (incrementando) sin tener el contador despejado. El contador pasará a 0x00 cuando pase su valor máximo de 8 bits (TOP = 0xFF).

En funcionamiento normal, el indicador de desbordamiento del temporizador/contador (TOV2) se establecerá en el mismo ciclo de reloj del temporizador a medida que el TCNT2 se convierta en cero. La bandera TOV2, en este caso, se comporta como un noveno bit, excepto que solo está configurada, no borrada. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el indicador TOV2, la resolución del temporizador se puede aumentar mediante software. No hay casos especiales a considerar en el modo Normal, se puede escribir un nuevo valor de contador en cualquier momento.

Independientemente del modo que se utilice, el programador debe recordar dos cosas:

1. El temporizador debe iniciarse seleccionando la fuente del reloj.
2. Si se utilizan interrupciones, deben estar habilitadas.

### **Ejemplo de temporizador de MCU AVR® de 8 bits**

Esta página ilustra varios métodos para configurar los temporizadores en un MCU AVR® de 8 bits. Se proporciona una descripción general de los temporizadores en un AVR antes de presentar el ejemplo paso a paso.


### **Requisitos para ejecutar los ejemplos prácticos**

#### **Requisitos de hardware**

1. ATmega328PB Xplained Mini placa
2. Micro-USB Cable

Tool

[About](#)
[Purchase](#)









**ATmega328PB Xplained Mini**  
 Evaluation Kit

[?](#)
[🛒](#)







The ATmega328PB Xplained Mini evaluation kit is a hardware platform for evaluating the Atmel ATmega328PB microcontroller. An external Debugger kit is NOT needed to run these labs. The ATmega328PB has a fully integrated embedded debugger on-board.

This hands-on exercise demonstrates how to set up the different timers.


### Software Tools

Tool	About	Installers			Installation Instructions
		Windows	Linux	Mac OSX	
 <b>Atmel® Studio</b> Integrated Development Environment					

### Software Tools

Tool	About	Installers			Installation Instructions
		Windows	Linux	Mac OSX	
 <b>Atmel® Studio</b> Integrated Development Environment					

### Additional Files

Files
 <a href="#">Xplained Board User Guide</a>

## Paso a paso

### Creación de proyectos

1

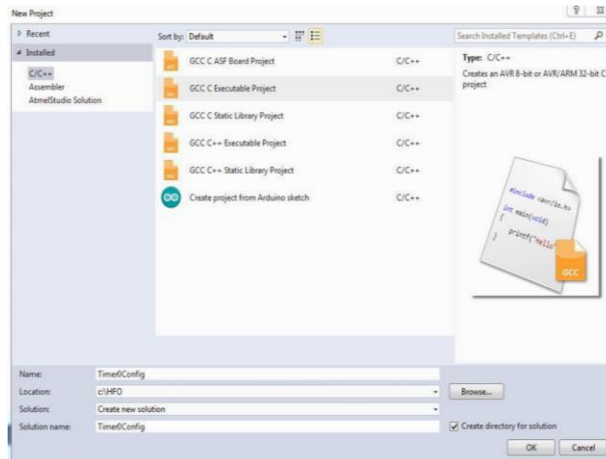
Abrir Atmel Studio 7

2

Seleccione **Archivo > nuevo proyecto de >**

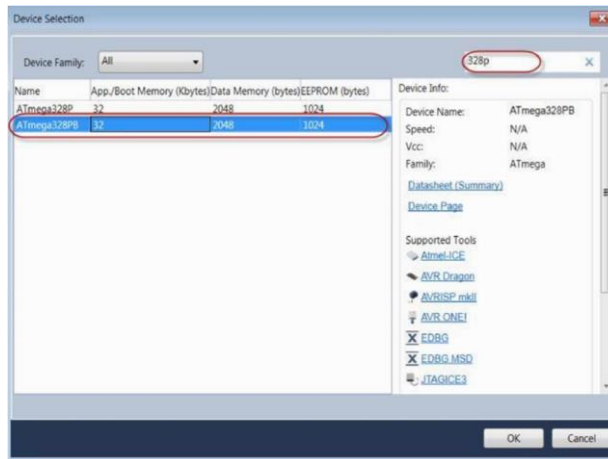
3

En la ventana **Nuevo proyecto**, seleccione **Proyecto ejecutable GCC C** y asigne el nombre del proyecto a "Timer0Config". Establezca la ubicación de los archivos de proyecto. (En este ejemplo se utiliza "C:\HFO\TimersConfigurations"). Haga clic en **Aceptar**.



4

En la barra de búsqueda de la ventana **Selección de dispositivos**, escriba "328p" y luego seleccione el dispositivo **Atmega328PB**, haga clic en **Aceptar**.



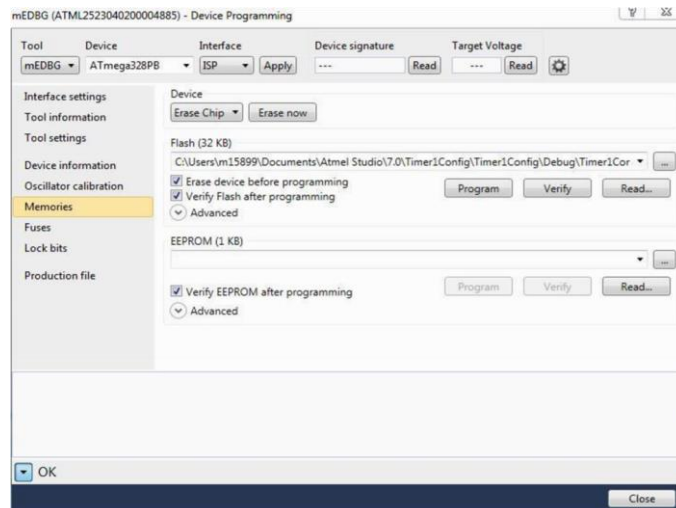
## Programación de la placa

5

Seleccione **Generar > solución de compilación** en el menú superior para compilar el código. Verá un mensaje "BUILD SUCCEEDED" en la ventana de salida de Studio 7.

6

Seleccione **Herramientas > Programación de dispositivos**, asegúrese de que la configuración sea como en la imagen a continuación y haga clic en **Aplicar**.



7

Seleccione **Memorias > programa** Espere hasta que el botón "Verificación de Flash... OK" para aparecer..

### Modificaciones del proyecto

avr/io.h y avr/interrupt.h deben ser "#included" en main.c para que este proyecto se compile y ejecute.

1. io.h proporciona la definición de todos los periféricos AVR.
2. interrupt.h es necesario para todas las aplicaciones que utilizan interrupciones.

8

### Modificación 1 - Parpadear un LED usando TC0

En este ejemplo, el temporizador 0 parpadea el LED conectado a PB5 cada 32 ms.

a

Modifique la función principal agregando el siguiente código:

```
1 int main(void)
2
3 {
4     //set RB5 as output
5     DDRB |= 1 << DDRB5;
6
7     //call TMR0 initialization function
8     init_TC0();
9
10    //enable interrupt
11    sei();
```

```

12
13 while (1)
14 {
15     //main loop
16 }
17 }

```

## b

Cree la función `init_TCO ()` en `main.c`. Agregue el código siguiente:

```

1 void init_TCO(void)
2 {
3
4     // Set the Timer Mode to CTC
5     TCCR0A |= (1 << WGM01);
6
7     // Set the value that you want to count to
8     OCR0A = 0xF9; //249
9
10    //Set the ISR COMPA vect
11    TIMSK0 |= (1 << OCIE0A);
12
13    // set prescaler to 1024 and start the timer
14    TCCR0B |= (1 << CS00) | (1 << CS02);
15 }

```

El código anterior hace lo siguiente:

1. Configura el temporizador 0 en modo CTC estableciendo el bit WGM correspondiente en el registro TCCR0A;
2. Establece el valor TOP en el registro OCR0A calculado utilizando la ecuación 1 para parpadear el LED a 30 kHz;
3. Habilita la interrupción del temporizador;
4. Establece el preescalador en 1024 configurando los bits CS00 y CS02 en el registro TCCR0B;

## c

Agregue la rutina de servicio de interrupción a `main.c`:

```

1 ISR (TIMER0_COMPA_vect)
2 {

```

```

3 //event to be executed every 32ms*MyTimerConstant
4 counter++;
5 if (counter == MyTimerConstant) {
6     counter = 0;
7     PORTB ^= 1 << PORTB5; //toggle LED on PORTB5
8 }
9 }

```

El uso de la constante *MyTimerConstant* y el *contador* de variables es opcional. Si se usa, cambiar el valor de MyTimer Constant alterará el tiempo que tarda el LED en parpadear.

**9**

### **Modificación 2 - Uso de TC1 en modo PWM para atenuar el LED**

En este ejemplo se utiliza TC1 para generar una señal PWM que se alimenta a través de un pin de E/S para accionar un LED. Cambiar el ciclo de trabajo de PWM resultará en un cambio en el brillo del LED.

**a**

Modifique main() agregando el siguiente código.

```

int main(void)
{
    //set direction of pin PB1 set as output
    DDRB |= 1 << DDRB1;

    init_TC1_pwm();

    //enable global interrupts
    sei();

    while (1)
    {
        //main loop
    }
}

```

Para configurar el bit de dirección del pin PB1, escriba el bit DDB1 en la lógica 1 en el registro DDRB. En la mini placa xplained ATmega328PB, el LED está conectado a PB5 y PWM se genera en PB1. Para ver la atenuación del LED, conecte un cable de PB5 a PB1 en la mini placa xplained ATmega328PB, como se muestra a continuación.

**b**

Cree la función `init_TC1_pwm ()` antes del bucle principal con el código siguiente:

```
1 void init_TC1_pwm(void)
2 {
3     //clear OCnA on compare match, BOTTOM (non-inverting mode)
4     TCCR1A = (1 << COM1A1);
5
6     //the counting sequence is determined by the setting of the waveform gener
7     TCCR1A |= (1 << WGM10);
8     TCCR1B |= (1 << WGM12);
9
10    //256 prescaler clock select bits
11    TCCR1B |= (0 << CS10) | (1 << CS12);
12
13    //enable interrupts
14    TIMSK0 |= (1 << OCIE1A);
15 }
```

`init_TC1_PWM()` realiza lo siguiente"

1. Establece los bits COMA1=1 y COMA0=0 (modo no inversor de la tabla de la hoja de datos Compare Output Mode, Fast PWM en el registro TCCR1A;
2. Configura bits 1:0 – WGM[1:0]:Waveform Generation Mode en consecuencia para Fast PWM, modo de 8 bits;
3. Verifica la configuración de la fuente de reloj y el preescalador que utilizará el temporizador/contador, que decide la frecuencia del PWM. La fuente de reloj interna dividida por 256 se configura escribiendo en los bits CS10 y CS12 en el registro TCCR1B;
4. Habilita la interrupción del temporizador;

**c**

Agregue la función ISR después del bucle principal para ejecutar el evento LED de atenuación:

```
1 ISR (TIMER1_COMPA_vect) // timer1 interrupt
2 {
3     duty_cycle--;
4     if (duty_cycle == 0) {
5         duty_cycle = 0xFF;
6     }
7     OCR1A = duty_cycle;
8
9 }
```

El registro OC1RA decide el ciclo de trabajo de PWM. Para atenuar el LED, disminuya gradualmente el ciclo de trabajo.

## 10

### Parpadear un LED usando TC2

Para este ejemplo, el temporizador 2 (TC2) está configurado para parpadear el LED conectado a PB5 en un período de 3 segundos.

#### a

Modifique la función principal agregando el siguiente código:

```
1 int main(void)
2 {
3     //set direction of PB5 as output
4     DDRB |= 1 << DDRB5;
5
6     /* Timer clock = I/O clock / 1024 */
7     TCCR2B = (1 << CS22) | (1 << CS20) | (1 << CS21);
8
9     /* Clear overflow flag */
10    TIFR2 = 1 << TOV2;
11
12    /* Enable Overflow Interrupt */
13    TIMSK2 = 1 << TOIE2;
14
15    // enable global interrupts
16    sei();
17
18    while (1)
19    {
20        // Main loop
21    }
22 }
```

1. Configura el bit de dirección del pin PB5, escribe el bit DDB5 en la lógica 1 en el registro DDRB;
2. Establece el preescalador en 1024 configurando los bits CS20, CS21 y CS22 en el registro TCCR2B;
3. Borra el indicador de desbordamiento: TOV2 se borra escribiendo uno lógico en el indicador.
4. Habilita interrupciones globales llamando a sei(); **b**

Habilite la interrupción del temporizador de desbordamiento;



Agregue la función ISR después del bucle principal para ejecutar el evento LED parpadeante:

Configure el registro PORTB para que parpadee el LED

```
1 ISR (TIMER0_COMPA_vect)
2 {
3     counterTimer2++;
4     if(counterTimer2 == MyTimer2Constant)
5     {
6         counterTimer2 = 0;
7         PORTB ^= 1 << PORTB5; //toggle led }
8 }
```

MyTimer2Constant es opcional. Esta constante se puede cambiar para alterar el tiempo que tarda el LED en parpadear.

## **Sensor de temperatura interno**

### **Sensor de temperatura interno AVR**

Algunos dispositivos AVR tienen un sensor de temperatura interno. Se puede utilizar para medir la temperatura central del dispositivo (no la temperatura ambiente alrededor del dispositivo). El voltaje medido tiene una relación lineal con la temperatura. La sensibilidad de voltaje es de aproximadamente 1 mV / ° C, la precisión de la medición de temperatura es de  $\pm 10^\circ \text{C}$ .

La medición de temperatura se basa en el sensor de temperatura en chip que está acoplado a un solo canal ADC de extremo. Seleccionar el canal ADC 8 escribiendo '1000' en ADMUX. MUX[3:0] habilita el sensor de temperatura. La referencia de voltaje interna de 1.1 V también debe seleccionarse para la fuente de referencia de voltaje ADC. Cuando el sensor de temperatura está habilitado, el convertidor ADC se puede utilizar en modo de conversión única para medir el voltaje sobre el sensor de temperatura.

#### Datos de medición de muestras

Temperature	-45°C	+25°C	+85°C
Voltage	242mV	314mV	380mV

#### Calibración

Los resultados de las mediciones de temperatura tienen errores de compensación y ganancia. La referencia de temperatura interna se puede corregir para estos errores realizando mediciones de calibración a una o dos temperaturas conocidas y ajustando los valores de salida. Esto puede resultar en mediciones de temperatura muy precisas, a veces tan precisas como  $\pm 2^\circ \text{C}$ . Se pueden encontrar más detalles en esta [nota de aplicación](#).

#### Configuración del ADC

La referencia de voltaje interno de 1.1 V debe seleccionarse para la fuente de referencia de voltaje ADC cuando se utiliza el sensor de temperatura interno. Al escribir "11" en los bits **REFS1** y **REFS0** del **registro ADMUX** se selecciona la referencia de voltaje interna de 1,1 V.

El ADC tiene múltiples canales de entrada y modos de operación. El **modo de conversión única** se puede utilizar para convertir la señal del sensor de temperatura conectada al canal 8. Para seleccionar el canal 8, escribiendo "1000" en los bits MUX3 a MUX0 se selecciona el canal 8 o el sensor de temperatura.

Una vez completada la conversión, el resultado se almacena en dos registros de datos ADC de 8 bits ADCH (8 bits más altos) y ADCL (8 bits inferiores). El resultado de 10 bits puede estar justificado a la izquierda o justificado a la derecha. Si el bit ADLAR se establece en un "1", entonces el resultado se deja ajustado a los 10 bits superiores de los dos registros. Si se establece en "0", el resultado ocupa los 10 bits inferiores de los dos registros. De forma predeterminada, cada bit está borrado y la palabra está justificada correctamente.

Esta instrucción de código establecerá los bits como se describe.  $\text{ADMUX} = (1 \ll \text{REFS1}) \mid (1 \ll \text{REFS0}) \mid (0 \ll \text{ADLAR}) \mid (1 \ll \text{MUX3}) \mid (0 \ll \text{MUX2}) \mid (0 \ll \text{MUX1}) \mid (0 \ll \text{MUX0});$

**Name:** ADMUX  
**Offset:** 0x7C  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	RW	RW	RW		RW	RW	RW	RW
Reset	0	0	0		0	0	0	0

**Bits 7:6 – REFSn: Reference Selection [n = 1:0]**  
 These bits select the voltage reference for the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

**Table 29-3 ADC Voltage Reference Selection**

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal $V_{ref}$ turned off
01	$AV_{CC}$ with external capacitor at AREF pin
10	Reserved
11	Internal 1.1V Voltage Reference with external capacitor at AREF pin

**Bits 3:0 – MUXn: Analog Channel Selection [n = 3:0]**  
 The value of these bits selects which analog inputs are connected to the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA on page 323 is set).

**Table 29-4 Input Channel Selection**

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

MUX[3:0]	Single Ended Input
0110	ADC6
0111	ADC7
1000	Temperature sensor
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	1.1V ( $V_{BG}$ )
1111	0V (GND)

### Configuración del reloj ADC y el tiempo de conversión

El ADC puede preescalar el reloj del sistema para proporcionar un reloj ADC que está entre 50 kHz y 200 kHz para obtener la máxima resolución. Si se requiere una resolución ADC de menos de 10 bits, entonces la frecuencia de reloj ADC puede ser superior a 200 kHz. A 1 MHz es posible alcanzar hasta ocho bits de resolución.

El valor del preescalador se selecciona con **bits ADPS** en **ADCSRA Register**. Por ejemplo; escribir "110" en el **registro ADCSRA** selecciona la división por 64 preescalador, lo que resulta en un reloj ADC de 125 KHz cuando se utiliza un reloj oscilador de 8 MHz.

Registro de control y estado de ADC A

Name: ADCSRA

Offset: 0x7A

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

**Bit 7 – ADEN: ADC Enable**  
Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

**Bit 6 – ADSC: ADC Start Conversion**  
In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.  
ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

**Bit 5 – ADATE: ADC Auto Trigger Enable**  
When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

**Bit 4 – ADIF: ADC Interrupt Flag**  
This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

**Bit 3 – ADIE: ADC Interrupt Enable**  
When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

**Bits 2:0 – ADPSn: ADC Prescaler Select [n = 2:0]**  
These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 29-5 Input Channel Selection

ADPS[2:0]	Division Factor
000	2
001	2

ADPS[2:0]	Division Factor
010	4
011	8
100	16
101	32
110	64
111	128

Iniciar una conversión

En el modo de conversión única, el **bit ADSC** en el **registro ADCSRA** debe establecerse en un estado lógico para iniciar la conversión ADC. Este bit permanece en la lógica alta mientras la conversión está en curso y es borrado por el hardware, una vez que se completa la conversión.

La primera conversión después de encender el ADC toma 25 ciclos de reloj ADC para inicializar el circuito analógico. Luego, para futuras conversiones, se necesitan 13 ciclos de reloj ADC (13.5 para conversiones activadas automáticamente).

## Ejemplo de sensor de temperatura interno AVR analógico a digital (ADC) de 8 bits

### Objetivo

Este proyecto práctico pasa por un ejemplo simple de lectura del sensor de temperatura en chip. Leer el sensor de temperatura puede ser un proyecto gratificante en sí mismo. Confirma que ha completado la compilación de software y la configuración de hardware del ADC, y que pudo programar el microcontrolador con éxito. Finalmente, el depurador se utiliza para ver la temperatura utilizando la ventana de salida de Studio 7.

El sensor de temperatura en chip está acoplado a un solo canal ADC8 de extremo. Selección del canal ADC8 escribiendo ADMUX. MUX[3:0] a '1000' habilita el sensor de temperatura. La referencia de voltaje interna de 1.1 V también debe seleccionarse para la fuente de referencia de voltaje ADC en la medición del sensor de temperatura. Cuando el sensor de temperatura está habilitado, el convertidor ADC se puede utilizar en modo de conversión única para medir el voltaje sobre el sensor de temperatura.

La sensibilidad de voltaje es de aproximadamente 1 mV / ° C, la precisión de la medición de temperatura es de  $\pm 10^\circ \text{C}$ .

### Materiales

#### Hardware Tools (Optional)

Tool	About	Purchase
 ATmega328PB Xplained Mini Evaluation Kit		

#### Software Tools

Installers					
Tool	About	Windows	Linux	Mac OSX	Installation Instructions
 Atmel® Studio Integrated Development Environment					

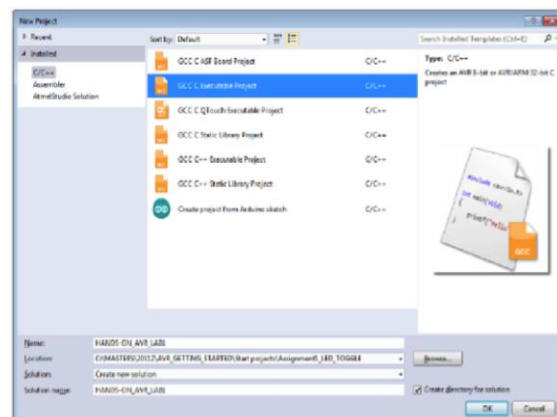


## Procedimiento

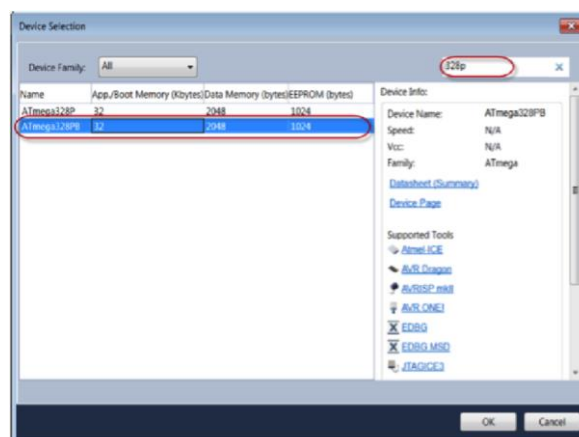
### 1

#### Tarea 1 - Creación de proyectos

1. Abrir Atmel Studio 7
2. Seleccione **Archivo > nuevo proyecto de >**
3. Seleccione GCC C Executable Project y asígnele el nombre **Project1 4.** Elija una ubicación para guardar el proyecto en su computadora



1. Aparecerá la ventana Selección de dispositivos. En la barra de búsqueda, ingrese **328P**, luego seleccione el dispositivo **Atmega328PB** y haga clic en Aceptar.



## 2

### Tarea 2 - Main.c

Este proyecto lee el sensor de temperatura interno, convierte el resultado a grados centígrados y luego almacena el resultado en **ADC Temperature Result**.

1) El archivo main.c es donde se agrega el código de la aplicación. El proyecto tiene un archivo main.c ya creado, pero solo contiene una instrucción while(1). Modifique main.c ingresando las líneas en el bloque de código gris a continuación.

#include <avr/io.h> se agrega automáticamente al archivo main.c cuando se genera. Esto siempre debe colocarse antes del bucle principal (vacío). El archivo de encabezado io.h llama al archivo iom328pb.h que define las definiciones de registro de ADC.

```
unsigned int Ctemp; unsigned int Ftemp;

int main(void)

{

    /* Setup ADC to use int 1.1V reference and select temp sensor
channel */

    ADMUX = (1<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (1<<MUX3) | (0<< MUX2) | (0<<MUX1) |
(0<<MUX0);

    /* Set conversion time to
112usec = [(1/(8Mhz / 64)) * (14 ADC clocks per conversion)] and enable the ADC*/

    ADCSRA = (1<<ADPS2) | (1<<ADPS1) | (1<<ADEN);

    /* Perform Dummy Conversion to complete ADC init */
    ADCSRA |= (1<<ADSC);

    /* wait for conversion to complete */ while ((ADCSRA &
(1<<ADSC)) != 0);

    /* Scan for changes on A/D input pin in an infinite loop */ while(1)
    {

        /* start a new conversion on channel 8 */
```



```

    ADCSRA |= (1<<ADSC);

    /* wait for conversion to complete */    while ((ADCSRA & (1<<ADSC)) != 0)

;

    /* Calculate the temperature in C */
    Ctemp = (ADC - 247)/1.22;
    Ftemp = (Ctemp * 1.8) + 32;
}    return -1;

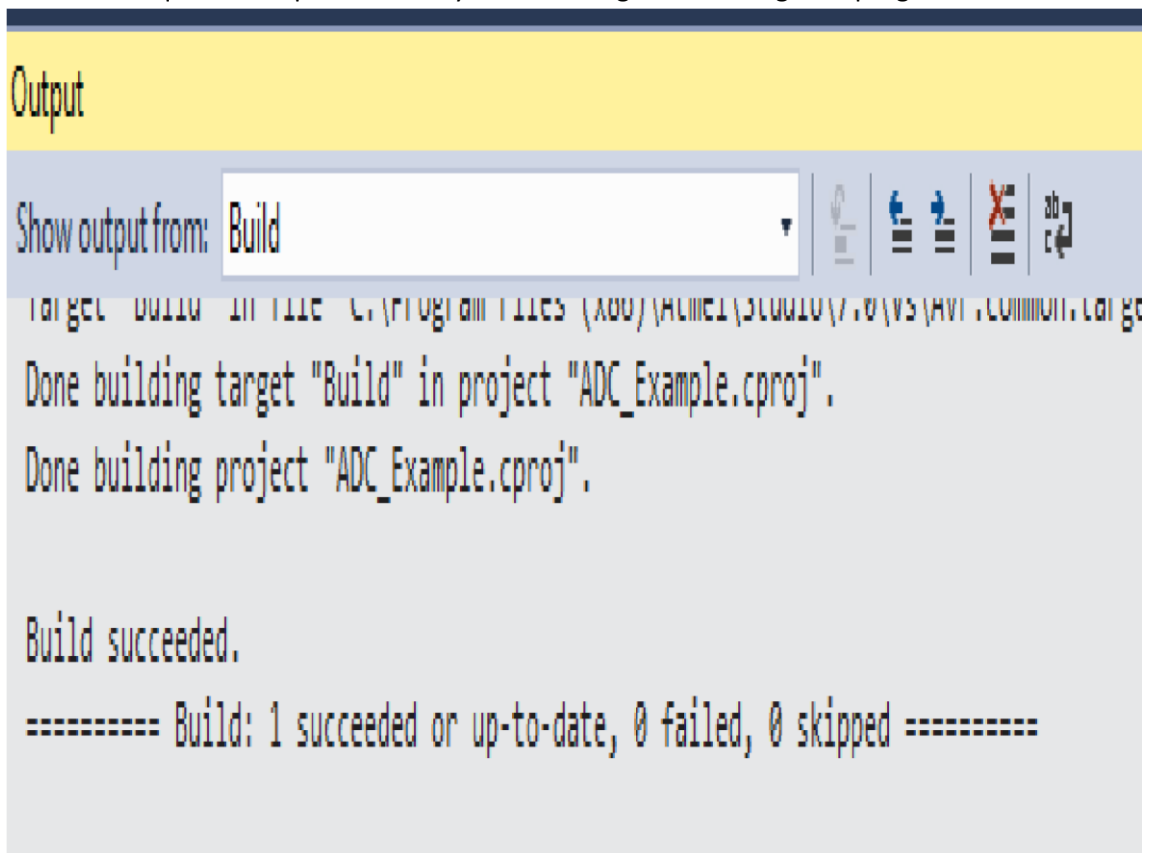
}

```

### 3

#### Tarea 3 - Generar proyecto

1. Seleccione **Generar > solución de compilación** en el menú Studio 7 para compilar el código. Verá un mensaje **De compilación correcta** en la ventana de salida. Si hay algún error, verifique main.c para ver si hay errores al ingresar el código del programa.



The screenshot shows the 'Output' window in Visual Studio. The 'Show output from:' dropdown is set to 'Build'. The output text is as follows:

```

target "Build" in file "C:\Program Files (x86)\Atmel\Studio7\7.0\VS\AVR\COMMON\target
Done building target "Build" in project "ADC_Example.cproj".
Done building project "ADC_Example.cproj".

Build succeeded.

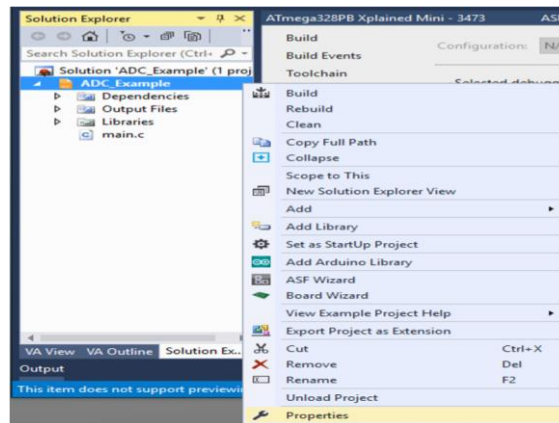
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====

```

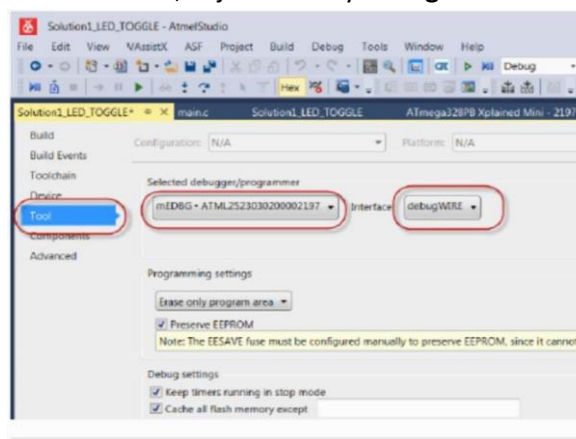
### 4

#### Tarea 4 - Programación de la placa Xplained

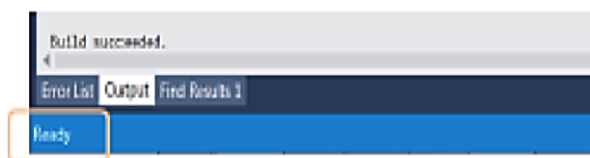
1. Conecte la placa Xplained al puerto USB del ordenador mediante el cable incluido.
2. En el área **Explorador de soluciones**, haga clic con el botón secundario en el nombre del proyecto y seleccione **Propiedades**.



1. En la selección del menú **Herramienta**, elija **mEDBG** y **debugWire** como interfaz.



1. Seleccione **Depurar > Iniciar sin depurar** en el menú de Studio 7. El proyecto compilará y luego programará la placa xplained con el código del proyecto junto con el control de depuración.



1. Studio 7 mostrará un mensaje **Listo** cuando se complete la programación.

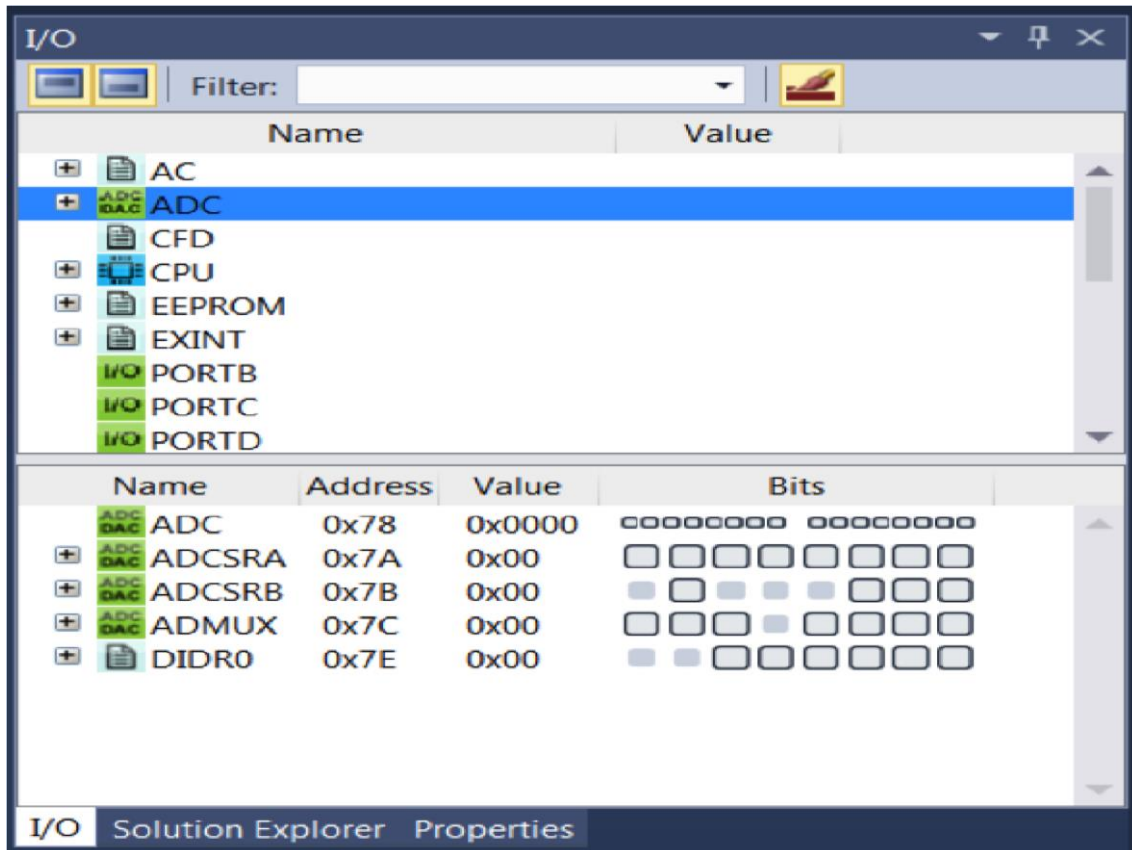
Si la placa Xplained no se conecta, puede haber una configuración de fusible que haga que esto ocurra.

#### Tarea 5 - Depuración

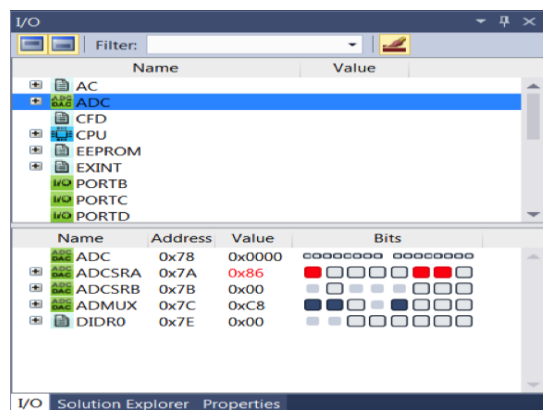
La depuración de un dispositivo es esencial para determinar cómo se puede estar ejecutando un programa.

1. Seleccione **Depurar > Iniciar depuración y Romper**. El proyecto se construirá y el programa se cargará en la placa Xplained

- Se abrirá la ventana Vista de E/S que muestra los distintos periféricos
- Haga clic en la selección de **ADC** para abrir la vista de E/S para el ADC



- Select **Debug > Step Over (F10)** to single-step through the program on the Xplained board. Monitor the ADC registers in the I/O View while single-stepping

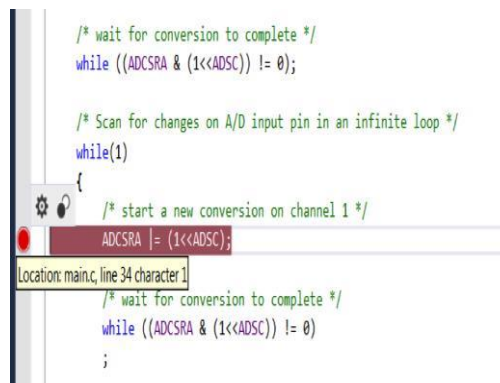


6

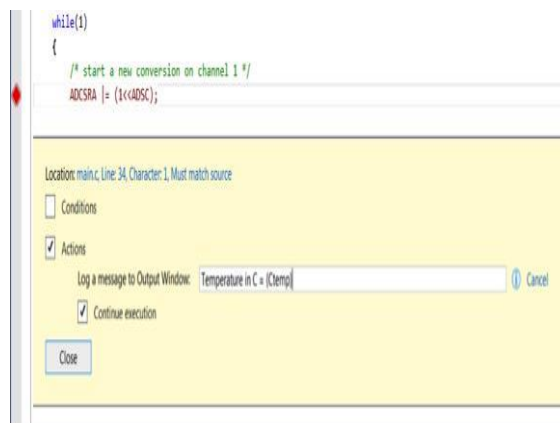
## Tarea 6 - Punto de interrupción y salida

1. Haga clic en el **menú Depurar > Romper todo** en el menú superior de Studio 7

2. Haga clic en el margen para habilitar un punto de interrupción en la línea de comandos en la instrucción ADCSRA dentro del bucle While



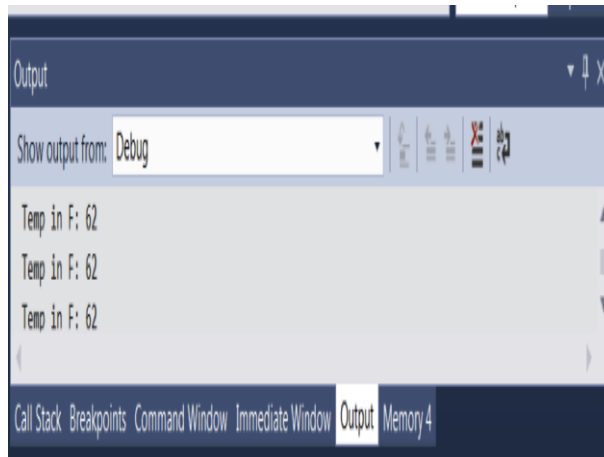
1. Mueva el mouse sobre el círculo rojo del punto de interrupción, para poder ver la opción emergente de configuración (símbolo de engranaje) y haga clic en ella



1. Dentro de la **ventana Configuración de punto de interrupción**, marque la casilla **Acciones**. Dentro de la "Registrar un mensaje en la ventana de salida", inserte lo siguiente: **Temperatura en C = {Ctemp}** y marque la **casilla Continuar ejecución**. Haga clic en **Cerrar**
2. Ingrese al modo de depuración haciendo clic en **Iniciar depuración y Romper** en el menú superior **Depurar**
3. Abra la **ventana de salida** seleccionando **Depurar > Salida de Windows >** para ver el valor de la variable de temperatura
4. Haga clic en **Depurar > Continuar** y ver la temperatura en la ventana Salida

## Resultados

La temperatura del chip se muestra en la ventana de salida. Presionarlo con un dedo calentará la lectura en un par de grados.



7

### Tarea 7 - Deshabilitar debugWIRE y Cerrar

El **fusible debugWire** debe restablecerse para programar la placa Xplained en el futuro. Mientras aún se ejecuta en modo de depuración, seleccione **Depurar > Deshabilitar debugWire y Cerrar**. Esto liberará el fusible debugWire.

### Análisis

El uso del ADC es bastante fácil y la función de depuración facilita el monitoreo de los resultados.

### Conclusiones

Este proyecto puede convertirse en la base para futuros proyectos relacionados con ADC.

## Funcionamiento de baja potencia

### Descripción general del funcionamiento de baja potencia de AVR® MCU

#### Descripción general de low power

El microcontrolador AVR® de 8 bits proporciona varios modos de suspensión y una compuerta de reloj controlada por software para adaptar el consumo de energía a los requisitos de la aplicación. Los modos de suspensión permiten que el microcontrolador apague los módulos no utilizados para ahorrar energía. Cuando el dispositivo entra en modo de suspensión, la ejecución del programa se detiene y se utiliza la interrupción o el restablecimiento para reactivar el dispositivo nuevamente. El reloj individual de los periféricos no utilizados se puede detener durante el funcionamiento normal o en reposo, lo que permite una administración de energía mucho más ajustada que los modos de suspensión solos.

Para alcanzar las cifras de potencia más bajas posibles hay un par de puntos a los que prestar atención. No es solo el modo de suspensión lo que define el consumo de energía, sino también el estado de los pines de E/S, la cantidad de módulos periféricos habilitados, etc.

El consumo de energía es proporcional al voltaje de funcionamiento, y para conservar la energía, debe considerarse el uso de un voltaje del sistema lo más bajo posible. Además, el consumo

también es directamente proporcional a la frecuencia del reloj, y si no se utilizan modos de suspensión, el dispositivo debe funcionar a la frecuencia más baja posible.

**Consejos y trucos para reducir la potencia en un AVR®**

1. Utilice el **registro de reducción de energía (PRR0)** para detener el reloj de los periféricos individuales no utilizados, lo que reduce el consumo de energía.

Name: PRR0  
Offset: 0x64  
Reset: 0x00  
Property: -

Bit	7	6	5	4	3	2	1	0
	PRTWI0	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

**Bit 7 – PRTWI0: Power Reduction TWI0**  
Writing a logic one to this bit shuts down the TWI 0 by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

**Bit 6 – PRTIM2: Power Reduction Timer/Counter2**  
Writing a logic one to this bit shuts down the Timer/Counter2 module in synchronous mode (AS2 is 0). When the Timer/Counter2 is enabled, operation will continue like before the shutdown.

**Bit 5 – PRTIM0: Power Reduction Timer/Counter0**  
Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

**Bit 4 – PRUSART1: Power Reduction USART1**  
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

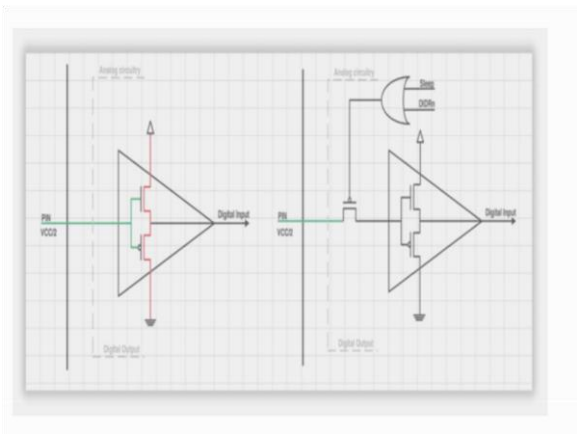
**Bit 3 – PRTIM1: Power Reduction Timer/Counter1**  
Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

**Bit 2 – PRSPI0: Power Reduction Serial Peripheral Interface 0**  
If using debugWIRE On-chip Debug System, this bit should not be written to one. Writing a logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

**Bit 1 – PRUSART0: Power Reduction USART0**  
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

**Bit 0 – PRADC: Power Reduction ADC**  
Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

1. Utilice el **registro de desactivación de entrada digital (DIDR)** para apagar los búferes de entrada digital no utilizados y detener la corriente de fuga.



Name: DIDR0  
Offset: 0x7E  
Reset: 0x00  
Property: -

Bit	7	6	5	4	3	2	1	0
	ADCD0	ADCD0	ADCD0	ADCD0	ADCD0	ADCD0	ADCD0	ADCD0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Name: DIDR1  
Offset: 0x7F  
Reset: 0x00  
Property: -

Bit	7	6	5	4	3	2	1	0
							AIN1D	AIN0D
Access							RW	RW
Reset							0	0

Bit 1 – AIN1D: AIN1 Digital Input Disable

Bit 0 – AIN0D: AIN0 Digital Input Disable

When this bit is written logic one, the digital input buffer on the AIN1/I/O pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/I/O pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

## Proyecto de ejemplo de bajo consumo AVR

### Objetivo

Esta página ilustra varios métodos para configurar un MCU AVR® de 8 bits para que funcione con un bajo consumo de energía. Con este ejercicio podrás:

1. Crear un proyecto y agregar código simple para parpadear un LED
2. Ejecute el proyecto y mida el consumo de energía
3. Realizar modificaciones en el código para reducir la potencia
4. Ejecute el proyecto modificado y observe un consumo de energía reducido **Materiales**



### Requisitos de hardware

1. ATmega328PB Xplained Mini placa
2. Kit de depurador de energía
3. Dos cables micro USB

1. Tres alambres de hembra a macho, y un alambre de macho a macho



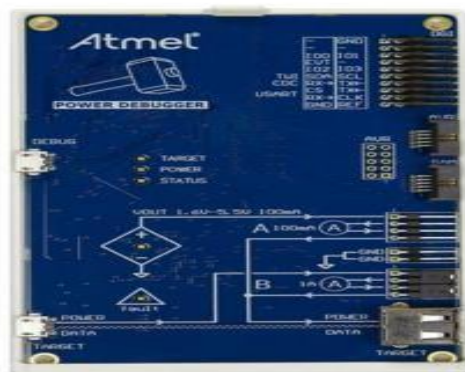
### ATmega328PB Xplained Baord

El **ATmega328PB Xplained Mini Evaluation Kit** es una plataforma de hardware para evaluar el microcontrolador Atmel ATmega328PB. NO se necesita un depurador externo para ejecutar estos ejercicios. El ATmega328PB tiene un depurador integrado totalmente integrado a bordo.



### Kit de depurador de energía

El Power Debugger es un depurador compatible con CMSIS-DAP que funciona con Atmel Studio v7.0 o posterior. El depurador de energía envía datos de medición de energía en tiempo de ejecución y depuración de aplicaciones al Visualizador de datos.



El depurador de potencia tiene dos canales de detección de corriente independientes para medir y optimizar el consumo de energía de un diseño. El canal 'A' es el canal recomendado para medir con precisión las corrientes bajas. El canal 'B' es el canal recomendado para medir corrientes más altas con menor resolución.

### Configuración de hardware

#### Conexión del depurador de energía a la placa Xplained ATmega328PB.

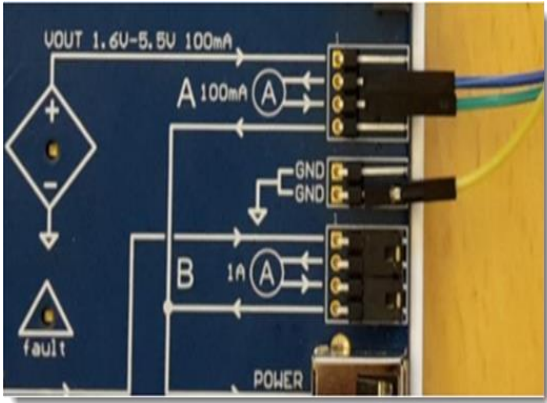
Los puertos de medición de corriente de canal 'A' y 'B' en la placa Power Debugger se representan con símbolos de amperímetro en la serigrafía. La fuente de alimentación de voltaje está conectada a la entrada del amperímetro, y la carga (objetivo) está conectada a la salida.



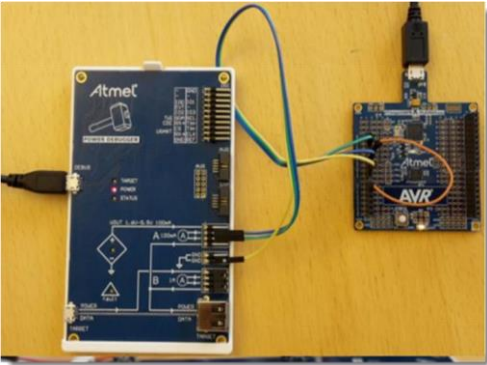
Con los siguientes pasos, el depurador de potencia mide el consumo en el núcleo AVR. Tabla 1-1 Power Debugger y **ATmega328PB Xplained Mini** conexión.

Power Debugger	ATmega328PB Xplained Mini
Port A input: Blue wire	5V
Port A output: Green wire	VCC
GND: Yellow Wire	GND

Figura 1-1 Depurador de potencia y conexión **ATmega328PB Xplained Mini**.



### Configuración de hardware



### Medición de la corriente en modo activo

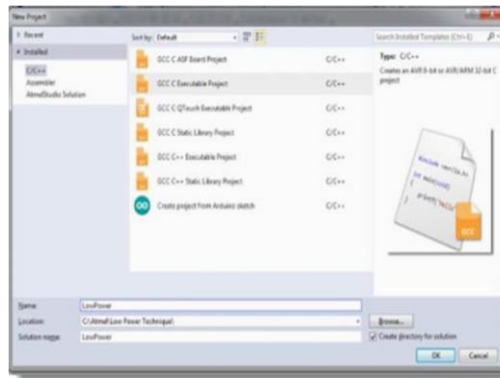
#### Procedimiento

#### A

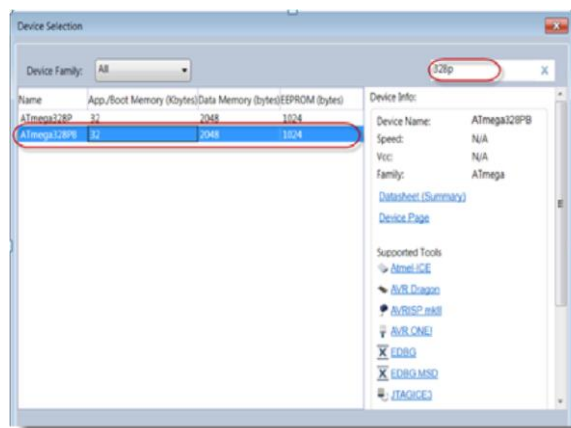
#### Creación de proyectos

1. Abrir Atmel Studio 7
2. Seleccione **Archivo > nuevo proyecto de >**

3. Seleccione **GCC C Executable Project** y asígnele el nombre **LowPower**.
4. Elija una ubicación para guardar el proyecto en el equipo.



1. Cuando aparezca la ventana Selección de dispositivos, introduzca 328P en la ventana de búsqueda y, a continuación, seleccione Atmega328PB. Haga clic en Aceptar.



1. Se creará un proyecto que contenga un bucle while(1) vacío en main().

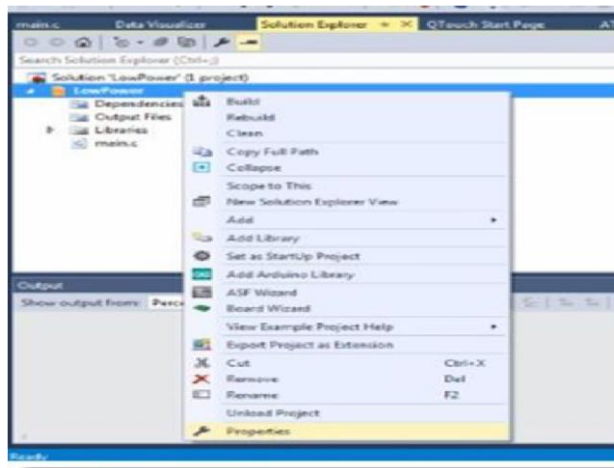
```

1 #include <avr/io.h>
2
3 int main(void)
4 {
5     /* Replace with your application code */
6     while (1)
7     {
8     }
9 }

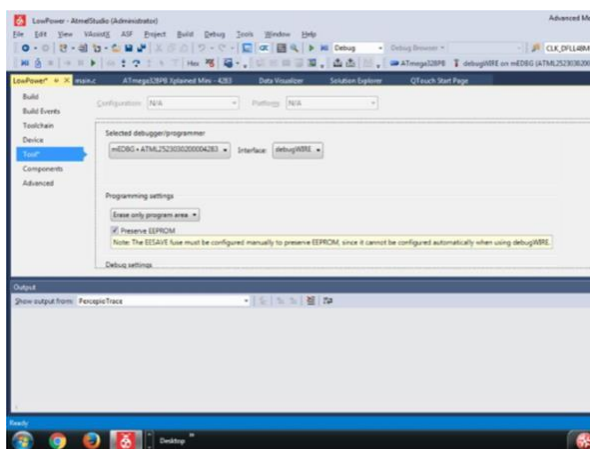
```

Seleccione **Ver > Explorador de soluciones** en la barra de menús.

En el Explorador de soluciones, haga clic con el botón secundario en el proyecto y seleccione **Propiedades**.



En **Herramienta**, seleccione **mEDBG** y **debugWIRE**



**B**

## Agregar código al proyecto

1. Agregue las dos instrucciones siguientes en `main()` para llamar a `TMRO_init` y establecer un pin de E/S como salida:

```
TMRO_init();
```

```
DDRB |= 1<<DDRB5; // Direction of pin PB5 set as Output
```

## Agregar la rutina de inicialización Timer 0

```

/*****
*****

TMR 0 Initialization

```

```
void TMRO_init( void ){
```

```

// enable timer overflow interrupt for both Timer0 and Timer1
*****
*****/
TIMSK0=(1<<TOIE0)
// set timer0 counter initial value to 0
TCNT0=0x00;
// start timer0 with /1024 prescaler
TCCR0B = (1<<CS02) | (1<<CS00);

// enable interrupts
sei();

}

```

1. Proporcione la rutina de servicio de interrupción TMRO al proyecto y haga que el LED parpadee a aproximadamente 1 Hz.

```

uint8_t count;
ISR(TIMERO_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}

```

El programa completo es el siguiente,

```

#include <avr/io.h>
#include "avr/interrupt.h"

uint8_t count;
/* Timer 0 Interrupt */
ISR(TIMERO_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}

int main(void)

```

```

{
}

{
  TMR0_init();

  DDRB |= 1<<DDR5; // Direction of pin PB5 set as Output
  /* Replace with your application code */
  while (1)
}

/*****
*****
***** TMR 0 Initialization
*****
*****/

```

## C

### Comprobar que el proyecto se ejecuta

1. Compile el proyecto y programe el dispositivo seleccionando **Depurar > Continuar**.

El LED parpadeará si el proyecto funciona correctamente

1. Asegúrese de que la depuración del proyecto haya finalizado seleccionando **Depurar > Deshabilitar debugWire** y, a continuación, **Cerrar**.

```

*****
void TMRO_init( void ){

// enable timer overflow interrupt for both Timer0 and Timer1
TIMSK0=(1 <<TOIE0) ;

// set timer0 counter initial value to 0
TCNT0=0x00;

// start timer0 with /1024 prescaler
TCCR0B = (1 <<CS02) | (1<<CS00);

// enable interrupts
sei();

}

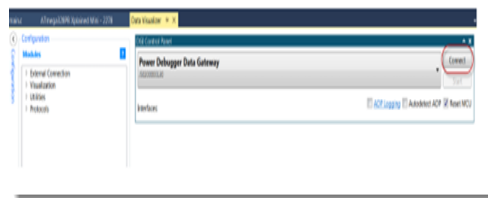
```

D

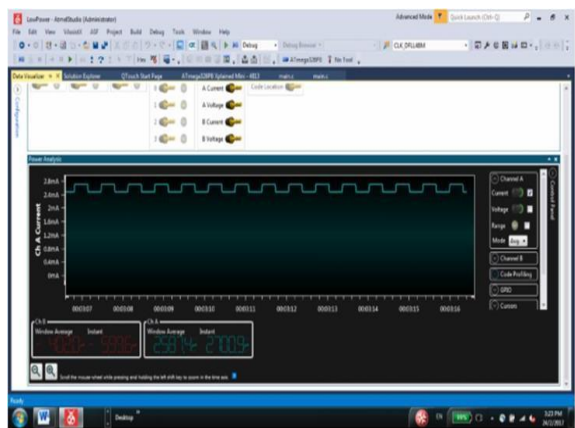
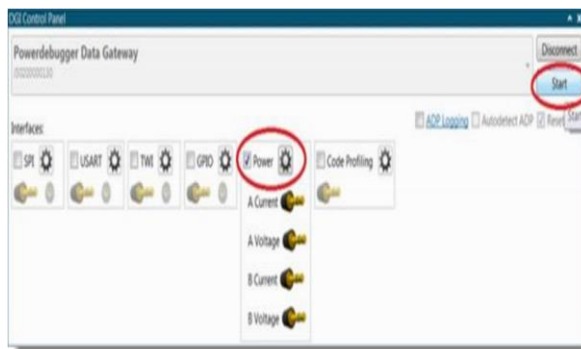
**Medir el consumo de energía en modo activo**

1. En Atmel Studio 7, abra el menú **Herramientas** > **Data Visualizer**

Power Debugger Data Gateway debe seleccionarse de forma predeterminada en el Panel de control de DGI, selecciónelo si no. Haga clic en **Conectar**.



1. Seleccione **Encendido** y **Start**



Apague la fuente de alimentación de destino a la placa Xplained y habilite la medición de potencia



1. Cierre la programación del dispositivo
1. Verifique que el consumo de corriente promedio sea de aproximadamente 2.6 mA

### Reducción del consumo de energía

Hay varios métodos que puede utilizar para reducir el consumo de energía de un microcontrolador AVR®. En este ejemplo se reduce la potencia en:

1. desactivar periféricos no utilizados
2. Detener la corriente de fuga en los pines de E/S digitales
3. Uso del modo de suspensión. Las técnicas de optimización del consumo de energía se implementan en la función `Optimize_power_consumption()` que se llama desde `main()`.

### Deshabilitar los búferes de entrada digital y el comparador analógico

Para los pines de entrada analógica, el búfer de entrada digital debe estar deshabilitado.

El búfer de entrada digital mide el voltaje en el pin y representa el valor como uno o cero lógico. Un nivel de señal analógica cercano a  $VCC/2$  en un pin de entrada puede causar un consumo de corriente adicional significativo. Si el pin se utiliza como pin analógico, no es necesario saber si el valor digital de la señal analógica sería uno o cero; estos pines digitales deben estar deshabilitados.

### Desactivar periféricos no utilizados

Deshabilite los periféricos no utilizados en la aplicación. El **registro de reducción de potencia (PRR0)** y (**PRR1**) puede detener el reloj en periféricos individuales para reducir el consumo de energía. Los recursos utilizados por el periférico permanecen ocupados cuando el reloj se detiene. En la mayoría de los casos, los periféricos deben desactivarse antes de que se detenga el reloj.

1. Incluya el archivo `<power.h>` en el programa principal

```
#include <avr/power.h>
```

2. Agregue el siguiente código a `optimize_power_consumption()`.

```
/* Power shutdown to unused peripherals*/
PRR0 = 0xDF;
PRR1 = 0x3F;
```

3. Agregue y #include para <wdt.h> a main.c.

**#include <avr/wdt.h>** 4. Agregue el siguiente código en optimize\_power\_consumption() para desactivar el **temporizador de vigilancia**.

```
/* Disable interrupts */
cli();
Wdt_reset();
MCUSR &= ~(1 << WDRF);
```

### Aplicar resistencias pull-up

Los pines no utilizados y no conectados consumen energía. La carga de energía innecesaria de los pines flotantes se puede evitar utilizando las resistencias de extracción internas del AVR en los pines no utilizados. Los pines de puerto se pueden configurar para introducir pull-ups estableciendo el bit **DDxn** en 0 y el bit **PORTxn** en 1. (donde **x** es PUERTO B, C, D, E y **n** es de 0 a 7)

```
/* Unused pins set as input pull up */
DDRB &= 0xE0;
DDRC &= 0xC9;
DDRD &= 0x03;
DDRE &= 0xF3;

PORTB |= ~(0xE0);
PORTC |= ~(0xC9);
PORTD |= ~(0x03);
PORTE |= ~(0xF3);
```

### Usar la función de suspensión

El modo de suspensión permite que la aplicación ahorre energía apagando los módulos no utilizados. El AVR proporciona varios modos de suspensión que permiten al usuario adaptar el consumo de energía a los requisitos de la aplicación.

1. **Incluya <avr/sleep.h>** archivo de encabezado de la biblioteca AVR en la parte superior del archivo.

2. Configure el modo de suspensión deseado en la función optimize\_power\_consumption () configurando el microcontrolador para que utilice el modo de suspensión **SLEEP\_MODE\_PWR\_DOWN** para un consumo de energía mínimo.

```
/* Set sleep mode */
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
```

3. Llame a la función sleep\_mode() desde main() para entrar en modo de suspensión.

```
while (1)
{
    sleep_mode();
}
```



```
}
```

### Uso de la interrupción de cambio de pin

Para activar el microcontrolador desde **SLEEP\_MODE\_PWR\_DOWN** se utiliza la **interrupción de cambio de pin**. El interruptor de la placa **ATmega328PB Xplained Mini** (SW0) está conectado a **PB7**. Pin **PB7** es la fuente de la interrupción de cambio de pin. Haga las siguientes adiciones a `main()`:

**#include avr/interrupt.h**

Configuración del bit PCINT7 y el registro PCICR:

```
PCMSK0 |= (1<<PCINT7); //Enable Pin Change Interrupt 7
PCICR |= (1<<PCIE0); //Enable the interrupt enable bit for PCINT
sei();
```

### Agregue la rutina de servicio de interrupciones:

Para habilitar las rutinas ISR, el nombre vectorial para el PCINT7 es ISR (PCINT0\_vect), definido en el archivo de encabezado del dispositivo.

```
ISR (PCINT0_vect)
{
    if (!(PINB & (1<<PINB7))) // if PINB7 is low (Switch pressed)
    {
        PORTB |= (1 <<PORTB5); // Turn ON LED
    }
    else
    {
        PORTB &= ~(1<<PORTB5); // Turn OFF LED
    }
}
```

### Código completado

Después de realizar los cambios anteriores, al código completo le debe gustar lo siguiente:

```
/*
 * lowpower2.c
 */

//#include <avr/io.h>

#include <avr/io.h>
#include "avr/interrupt.h"
#include "avr/wdt.h"
```

```

PORTB |= ~(0xE0);
PORTC |= ~(0xC9);
PORTD |= ~(0x03);
PORTE |= ~(0xF3);

/* Watchdog Timer OFF */

/* Disable interrupts */
cli();
/* Reset watchdog timer */
wdt_reset();
/* Clear WDRF in MCUSR */
MCUSR &= ~(1 << WDRF);
/* Turn off WDT */
WDTCSR = 0x00;

/* Set sleep mode */
set_sleep_mode(SLEEP_MODE_PWR_DOWN);

#include "avr/sleep.h"

#include <avr/power.h>
#include <avr/interrupt.h>
// #include <util/delay.h>

void optimize_power_consumption(void){

/* Disable digital input buffer on ADC pins */
DIDR0 = (1 << ADC5D) | (1 << ADC4D) | (1 << ADC3D) | (1 << ADC2D) | (1 << ADC1D) | (1 << ADC0D);
DIDR0 |= 0xC0; /* ADC7D and ADC6D are undefined in header file so set bits this way */

/* Disable digital input buffer on Analog comparator pins */
DIDR1 |= (1 << AIN1D) | (1 << AIN0D);
/* Disable Analog Comparator */
ACSR |= (1 << ACD);

/* Power shutdown to unused peripherals */
PRR0 = 0xDF;
PRR1 = 0x3F;
/* Unused pins set as input pull up */
DDRB &= 0xE0;
DDRC &= 0xC9;
DDRD &= 0x03;
DDRE &= 0xF3;

```

```

}

/*****
*****

TMR 0 Initialization
*****/

void TMR0_init( void ){

    // enable timer overflow interrupt for both Timer0 and Timer1
    TIMSK0=(1<<TOIE0) ;

    // set timer0 counter initial value to 0
    TCNT0=0x00;

    // start timer0 with /1024 prescaler
    TCCR0B = (1<<CS02) | (1<<CS00);

    // enable interrupts
    sei();

}

uint8_t count;
/* Timer 0 Interrupt */
ISR(TIMERO_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}

ISR (PCINT0_vect)
{
    if (!(PINB & (1<<PINB7))) // if PINB7 is low (Switch pressed)
    {
        PORTB |= (1<<PORTB5); // Turn ON LED
    }
    else
    {
        PORTB &= ~(1<<PORTB5); // Turn OFF LED
    }
}

int main(void)
{

```

```

TMR0_init();

DDRB |= 1<<DDR5; // Direction of pin PB5 set as Output
DDRB &= ~(1<<DDB7); //Set PORTB7 as input

optimize_power_consumption();

PCMSK0 |= (1<<PCINT7); //Enable Pin Change Interrupt 7
PCICR |= (1<<PCIE0); //Enable the interrupt enable bit for PCINT
sei();

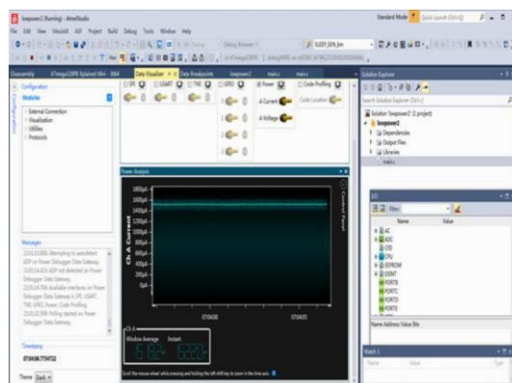
//set_sleep_mode(SLEEP_MODE_PWR_DOWN);
//sleep_enable();
//sleep_cpu();

/* Replace with your application code */
while (1)
{
    sleep_mode();
}
}

```

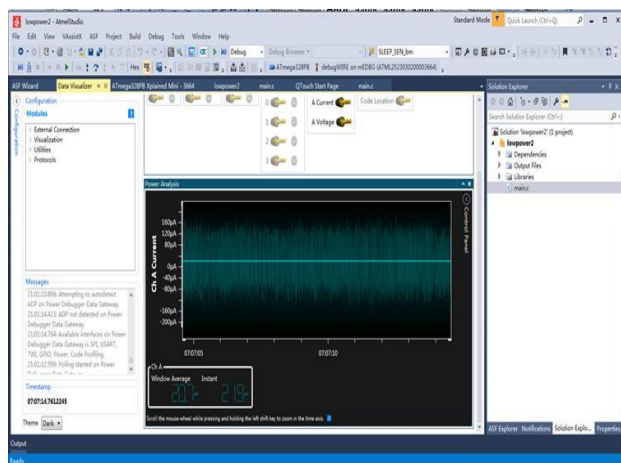
**Programa la aplicación y mida el consumo de energía.**

1. Programe el código seleccionando **Depurar > Continuar**.
2. Espere hasta que la aplicación esté programada y el mensaje en la parte inferior de la ventana aparezca como **En ejecución**.
3. Salga del depurador seleccionando **Depurar > Deshabilitar debugWire** y luego **Cerrar**.
4. Abra la **ventana de DataVisualizer** y verifique el consumo de energía como se muestra a continuación.



El consumo de corriente observado debería reducirse a alrededor de 1,52 mA.

1. Apagar el interruptor de alimentación de destino: **Herramientas > configuración de Programación de dispositivos > Herramientas**



El consumo actual debería haberse reducido a aproximadamente 20,7  $\mu\text{A}$ .

Tenga en cuenta que el temporizador no activa el dispositivo desde el modo DE SUSPENSIÓN. La aplicación está configurada para salir del modo de suspensión cuando se produce la **interrupción de cambio de pin**.

## Conclusiones

En este ejemplo se demostraron varias técnicas diferentes para reducir el consumo de energía de una aplicación AVR. El consumo de energía se puede reducir significativamente mediante:

1. Diseño inteligente
2. Uso de modos de suspensión
3. Apagar periféricos no utilizados

En este ejemplo se utilizó el **visualizador de datos Atmel Studio 7** y la **placa Power Debugger** para medir la potencia.

## Capacitación en ADC y optimización de energía para MCU tinyAVR® y megaAVR®

### Introducción:

Este tutorial contiene cinco aplicaciones prácticas para la conversión de datos ADC, con el consumo de corriente medido para cada aplicación. El tutorial comienza con una sencilla aplicación de conversión de ADC. En las siguientes aplicaciones, se introducen diferentes técnicas para demostrar cómo se puede reducir el consumo de corriente en **los MCU diminutos de®** las series 0 y 1 de AVR y **megaAVR®** de la serie 0.

Este tutorial también muestra cómo usar Atmel START para comenzar con el desarrollo de aplicaciones ADC de dispositivos **AVR®**. Las aplicaciones ADC se han desarrollado paso a paso en Atmel Studio. Este tutorial se ha desarrollado en el kit de evaluación ATtiny817 Xplained Pro, pero debería ser aplicable a todos los dispositivos diminutos de las series 0 y 1 de AVR y megaAVR de la serie 0.

Los proyectos de solución para cada una de las asignaciones están disponibles en el [explorador de ejemplo START de Atmel](#). En la categoría "Introducción", busque ADC y Power Optimization Solution (1-5). Los enlaces directos a los proyectos de ejemplo relevantes se proporcionan en las descripciones de las asignaciones a continuación.

1. [ADC y optimización de energía: manual tutorial práctico](#)

**Requisitos previos de hardware**

1. Kit de evaluación ATtiny817 Xplained Pro
2. Cable micro-USB (Tipo-A/Micro-B)
3. Un potenciómetro
4. Tres cables macho a hembra 5. Conexión a Internet

**Requisitos previos de software**

1. Estudio Atmel 7.0
2. Navegador. La lista de navegadores compatibles se puede encontrar aquí: [INICIAR navegadores compatibles](#)

**Tiempo estimado de finalización: 120 minutos**

**Tarea 1: Conversión de ADC con la aplicación de impresión USART**

En esta asignación, Atmel Studio se utiliza para desarrollar una aplicación utilizando controladores ADC y USART de Atmel START. El ADC está configurado para ejecutarse en modo de conversión única y se conecta un potenciómetro al pin de entrada del ADC para estudiar la funcionalidad del ADC. Los datos ADC se envían a través de USART al terminal integrado en el visualizador de datos de Atmel Studio. En Data Visualizer, el consumo actual de la aplicación se analiza utilizando el Power Analyzer integrado.

**Asignación 2: Las interrupciones RTC desencadenan la impresión ADC y USART**

En esta asignación, se utiliza el módulo Contador de tiempo real (RTC). La interrupción de desbordamiento RTC se utiliza para desencadenar una conversión de ADC cada medio segundo. La interrupción ADC Result Ready (RESRDY) desencadena una impresión del resultado ADC en el terminal USART. Cuando no se activa la interrupción de desbordamiento rtc, el dispositivo se mantiene en modo de espera para reducir el consumo de energía. Atmel START se utiliza para agregar el módulo RTC y configurar los controladores RTC, ADC, CPUINIT y SLEEPCTRL. Un proyecto de Atmel Studio se regenera después.

**Asignación 3: Optimización de energía en pines de E/S**

En esta asignación, el búfer de entrada digital en los pines de E/S se desactiva para reducir el consumo de corriente. El consumo de corriente se reduce aún más cuando el pin USART TX se configura como un pin de alta impedancia durante ningún período de transmisión de datos. Aquí

se utilizan los mismos controladores y configuraciones de la asignación anterior. Atmel Studio se utiliza para desarrollar aún más el código.

#### **Asignación 4: Conversión de ADC mediante el modo de comparación de ventanas**

En esta asignación, la interrupción lista para el resultado de ADC se reemplaza por la interrupción WCMP de ADC, para desencadenar una transmisión USART. En este caso, el resultado de ADC, que está por debajo del valor de umbral de la ventana de ADC, desencadena la transmisión USART. Los resultados de ADC, que están por encima del valor de umbral de ventana, se ignoran y no desencadenan ninguna transmisión USART. Atmel START se utiliza para reconfigurar el módulo ADC y el proyecto Atmel Studio se actualiza con la nueva configuración.

#### **Asignación 5: Sistema de eventos (EVSYS) utilizado para reemplazar el controlador de interrupciones RTC**

En esta asignación, el sistema de eventos con la señal de evento de desbordamiento RTC, en lugar de la interrupción de desbordamiento RTC, se utiliza para desencadenar una conversión ADC. El sistema de eventos permite la señalización directa de periférico a periférico. Permite un cambio en un periférico (el Generador de eventos) para desencadenar acciones en otros periféricos (los Usuarios de eventos) a través de canales de eventos sin usar la CPU. Una ruta de canal puede ser asincrónica o sincrónica con el reloj principal.

### **Resumen**

Este tutorial contiene cinco aplicaciones prácticas que realizan la conversión de datos ADC, con el consumo de corriente medido para cada aplicación. Comienza con una aplicación de conversión ADC simple y se introducen diferentes técnicas para demostrar cómo se puede reducir el consumo actual. Esta es una base útil para desarrollar futuras aplicaciones ADC con requisitos de consumo actuales específicos.

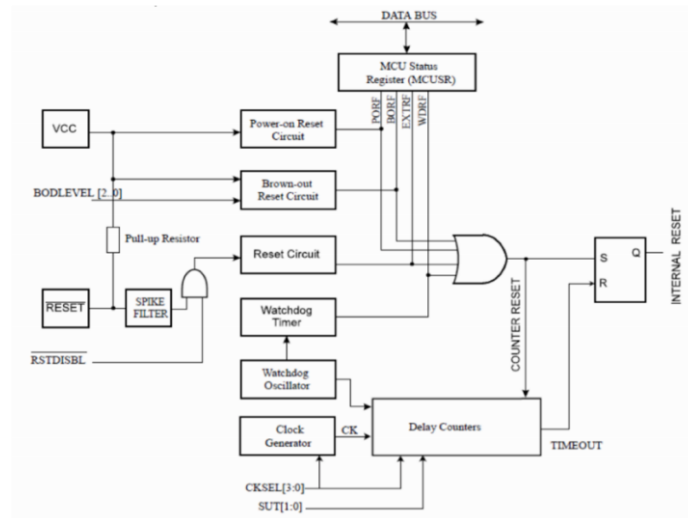
## **RESTABLECER FUENTES**

### **Fuentes de restablecimiento de AVR**

El dispositivo AVR tiene cuatro fuentes de restablecimiento:

1. **Restablecimiento de encendido:** el microcontrolador (MCU) se restablece cuando el voltaje de alimentación es inferior al umbral de restablecimiento de encendido (VPOT).
2. **Restablecimiento externo:** el MCU se restablece cuando hay un nivel bajo en el pin RESET durante más tiempo que la longitud mínima del pulso.
3. **Restablecimiento del sistema Watchdog:** el MCU se restablece cuando expira el período del temporizador Watchdog y el modo Watchdog System Reset está habilitado.

4. **Restablecimiento de salida marrón:** la MCU se restablece cuando la tensión de alimentación  $V_{CC}$  es menor que la restablecimiento de salida marrón.



### Registro de estado de MCU (MCUSR)

Para hacer uso de los indicadores de restablecimiento para identificar una condición de restablecimiento, el usuario debe leer y luego restablecer el MCUSR lo antes posible en el programa. Si el registro se borra antes de que se produzca otro restablecimiento, se puede encontrar el origen del restablecimiento examinando los indicadores de restablecimiento.

**Name:** MCUSR

**Offset:** 0x54

**Reset:** 0x00

**Property:** When addressing as I/O Register: address offset is 0x34

Bit	7	6	5	4	3	2	1	0
					WDRF	BORF	EXTRF	PORF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

**Bit 3 – WDRF: Watchdog System Reset Flag**

This bit is set if a Watchdog System Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

**Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

**Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

**Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a '0' to it.

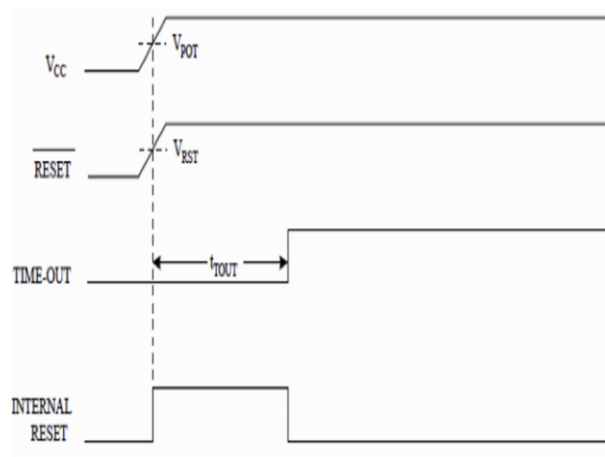
### Restablecimiento de encendido (POR)

Un pulso POR es generado por un circuito de detección en chip. El POR se activa siempre que  $V_{CC}$  esté por debajo del nivel de detección. El circuito POR se puede utilizar para activar el reinicio de arranque, así como para detectar una falla en el voltaje de alimentación.

Un circuito POR garantiza que el dispositivo se restablezca desde el encendido. Alcanzar el voltaje umbral de restablecimiento de encendido invoca el contador de retardo, que determina cuánto tiempo se mantiene el dispositivo en Restablecer después de que  $V_{CC}$  suba.



La señal de reinicio se activa de nuevo, sin demora, cuando  $V_{CC}$  disminuye por debajo del nivel de detección.

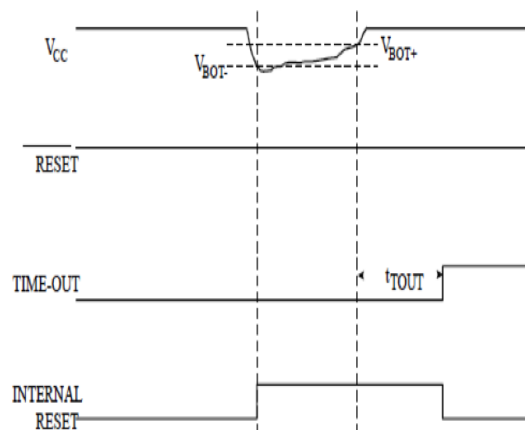


### Detección de apagado (BOD) y restablecimiento de apagado (BOR)

El dispositivo tiene un circuito BOD en chip para monitorear el nivel  $V_{CC}$  durante la operación comparándolo con un nivel de disparo fijo. El nivel de disparo para la DBO puede ser seleccionado por los fusibles BODLEVEL.

El circuito BOR tiene histéresis en el nivel de detección. El circuito BOD solo detectará una caída en  $V_{CC}$  si el voltaje permanece por debajo del nivel de disparo ( $V_{BOT-}$ ) durante más tiempo que  $t_{BOD}$ . Cuando eso ocurre, el BOR se activa inmediatamente.

Cuando  $V_{CC}$  aumenta por encima del nivel de activación ( $V_{BOT+}$  en la siguiente figura), el contador de retardo inicia el MCU después de que haya expirado el período de tiempo de espera  $t_{TOUT}$ .



### Temporizador watchdog (WDT)

El WDT se ejecuta independientemente del resto del sistema, lo que provoca que el sistema se restablezca cada vez que se agota el tiempo de espera. Sin embargo, el software de la aplicación debe asegurarse de que el tiempo de espera nunca se produzca restableciendo el WDT periódicamente, siempre y cuando el software esté en un estado de mantenimiento conocido. Si el sistema se bloquea o la ejecución del programa está dañada, el WDT no recibirá su

restablecimiento periódico y, finalmente, agotará el tiempo de espera y provocará un restablecimiento del sistema.

El WDT mejorado en algunos dispositivos AVR también tiene la capacidad de generar interrupciones en lugar de restablecer el dispositivo. Dado que el WDT se ejecuta desde su propio reloj independiente, se puede usar para activar el AVR desde todos los modos de suspensión. Esto lo convierte en un temporizador de activación ideal, que se combina fácilmente con la operación ordinaria como fuente de restablecimiento del sistema. La interrupción también se puede utilizar para obtener una advertencia temprana de un próximo restablecimiento del sistema Watchdog para que los parámetros vitales se puedan respaldar en la memoria no volátil.

### **Detección de fallos de reloj (CFD)**

El CFD permite al usuario monitorear el oscilador de cristal de baja potencia o la señal de reloj externa (XOSC). El XOSC es monitoreado por el circuito CFD que opera con el oscilador interno de 128kHz. CFD monitorea el reloj XOSC y si falla, cambiará automáticamente a un reloj RC interno seguro. Cuando se produce un encendido o un restablecimiento externo, el dispositivo volverá al reloj XOSC y continuará monitoreando el reloj XOSC en busca de fallas.

El reloj seguro se deriva del reloj interno del sistema RC de 8MHz. Esto permite configurar el reloj seguro para satisfacer las necesidades a prueba de fallos de la aplicación.

### **Proyecto de ejemplo de restablecimiento de orígenes de AVR®**

#### **Objetivo**




Este proyecto pasa por varias condiciones de reinicio diferentes: *Power On Reset (POR)*, *Brown Out Reset (BOR)* y *Watch Dog Timer (WDT)*, y muestra cómo funciona cada una en una placa **Xplained de 328PB**. Se muestra que algunos circuitos externos producen una entrada de voltaje variable, pero también funcionará una fuente de alimentación ajustable.

---

Para obtener más detalles sobre las **fuentes de restablecimiento de AVR®**, visite [Descripción general de fuentes de restablecimiento de AVR.](#)







## Material

**Hardware Tools (Optional)**






Tool	About	Purchase
 ATmega328PB Xplained Mini Evaluation Kit		

**Software Tools**

Installers


Tool	About	Windows	Linux	Mac OSX	Installation Instructions
 Atmel® Studio Integrated Development Environment					

**Exercise Files**

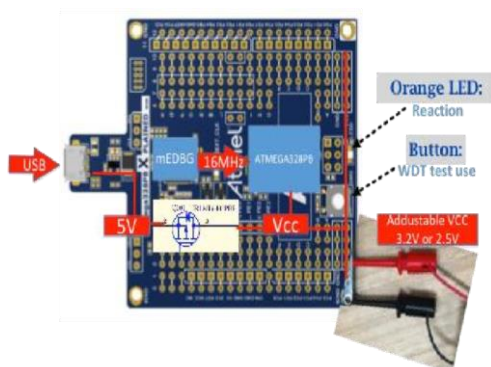
File	Download			Installation Instructions
	Windows	Linux	Mac OSX	
 Project and Source Files				

**Additional Files**

Files

-  328PB Xplained Mini Board User-Guide

## Connection Diagram

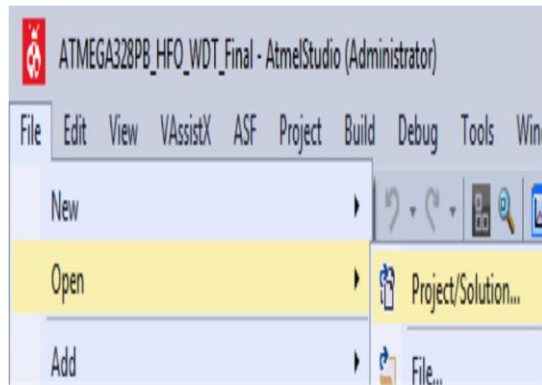


## Procedimiento

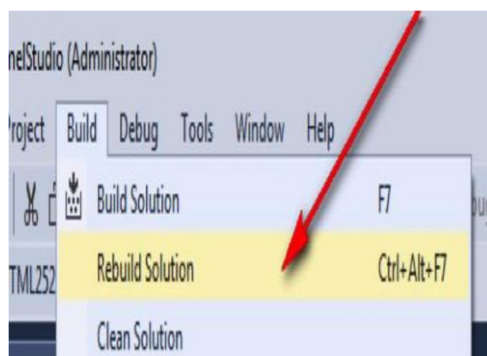
1

### Tarea 1 - Abrir y compilar proyectos

1. Descargue los archivos de origen del proyecto de la sección Archivos de ejercicio anterior y descomprimalos en su computadora.
2. Abrir Atmel Studio 7
3. Seleccione **Archivo > Abrir > proyecto/solución**



- Select ATMEGA328PB\_HFO\_WDT\_Final.atsln from the downloaded project files.
- From the Build menu select 'Rebuild Solution'.

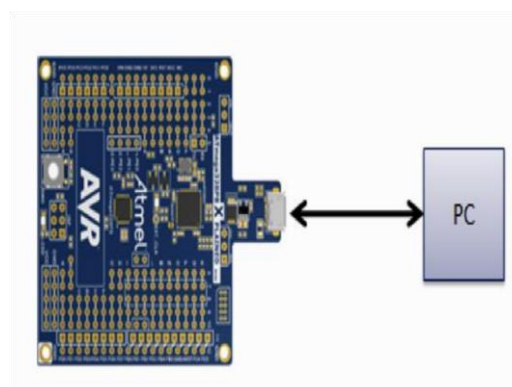


Seleccione 'GCC C Executable Project' y asígnele el nombre Project1. Elija una ubicación para guardar el proyecto en el equipo.

**2**

## Tarea 2 - Preparación de la Junta

Asegúrese de que el cable USB esté conectado entre la placa y el PC.



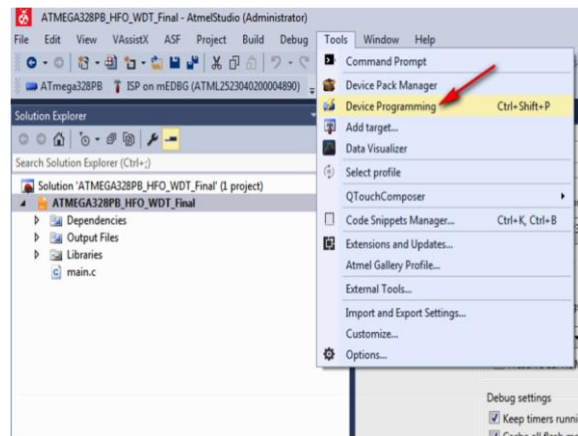
**3**

## Tarea 3 - Configuración del programador

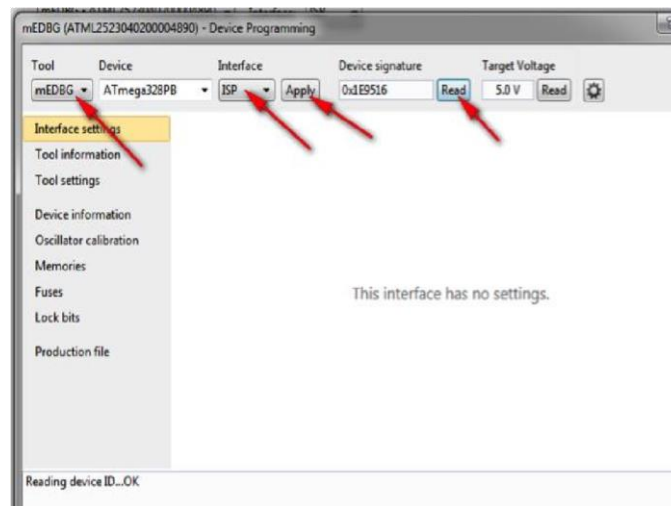
Si debugWire está habilitado, desactívelo.

Deshabilitar debugWire (DWEN) ►

1. Haga clic en **Herramientas > programación de dispositivos:**



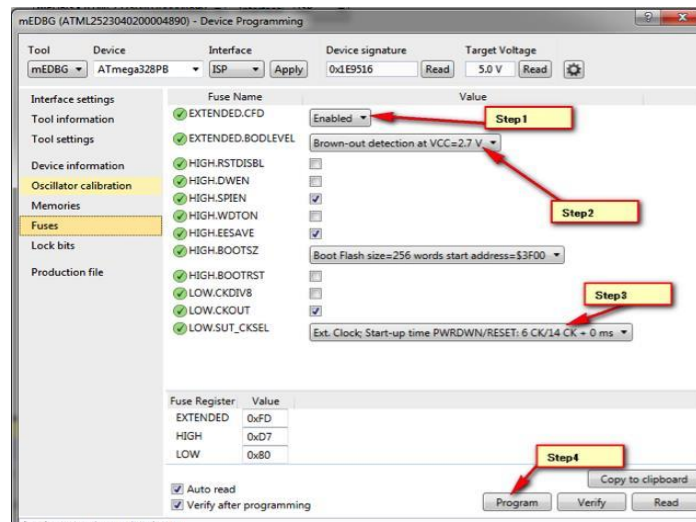
1. Haga clic en los botones que están marcados como flechas rojas en la captura de pantalla.



**4**

#### **Tarea 4 - Programación de los bits de fusible**

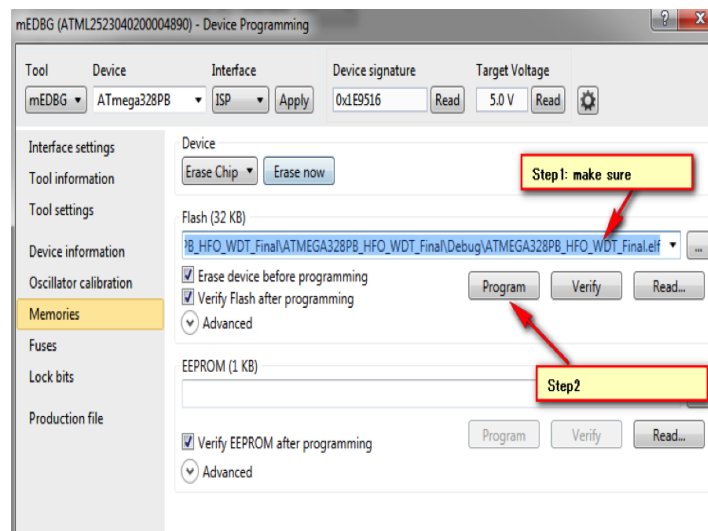
1. Seleccione las opciones 'Fusibles' en la barra de opciones izquierda y establezca los bits como se muestra en la figura, lo que habilitará CFD, el umbral de BOD como 2.7 V y el reloj externo.



5

## Tarea 5 - Programación de dispositivos

Seleccione las opciones 'Memorias' en la barra de opciones izquierda y finalice los pasos como se muestra en la figura, que programa el archivo binario compilado en ATMEGA328PB.



6

## Tarea 6 - Código del proyecto

Aquí está el código completo al que hacemos referencia. También puede descargar esto en la sección ejercicios.

```
/*
* ATMEGA328PB_HFO_WDT_Final.c
*
* Created: 2017/03/08 17:15:35
* Author : A17582
```

```
*/
```

```
#define F_CPU 8000000UL
```

```
#include <avr/io.h>
```

```
#include <avr/interrupt.h>
```

```
#include <avr/wdt.h>
```

```
#include <util/delay.h>
```

```
//initialize watchdog void
```

```
WDT_Init(void)
```

```
{
```

```
    //disable interrupts    cli();
```

```
    WDTCSR = (1<<WDCE)|(1<<WDE);           // Enable configuration change.
```

```
    WDTCSR = (1<<WDIE)|                     // Enable Watchdog Interrupt Mode.
```

```
    (1<<WDCE)|(1<<WDE);                     // Enable Watchdog System Reset Mode if  
unintentionally enabled.
```

```
    (0<<WDP3)|(1<<WDP2)|(1<<WDP1)|(1<<WDP0); // Set Watchdog Timeout period to  
4.0 sec.
```

```
    //Enable global interrupts    sei();
```

```
}
```

```
//Watchdog timeout ISR
```

```
ISR(WDT_vect)
```

```
{
```

```
    //Burst of 0.1Hz pulses    for (uint8_t
```

```
i=0;i<4;i++)
```

```
{
```

```
    //LED OFF
```

```
    PORTB &= ~(1<<PINB5);           // Set PORTB5 Low
```

```

        _delay_ms(80);

        //LED ON

        PORTB |= (1 << PINB5);          // Set PORTB5 On

        _delay_ms(20);
    }
}

#define B0RFB0 2
#define P0RFB0 0

int main(void)
{
    unsigned char i;
    DDRB |= (1 << PINB5);          // Set PORTB5 as output ,
    DDRB &= ~(1<<PINB7);          //Set PORTB7 as input

    if(MCUSR & 1){      MCUSR=0;
for ( i=0;i<4;i++)

    {
        //LED OFF
        PORTB &= ~(1 << PINB5);      // Set PORTB5 Low
        _delay_ms(300);
        //LED ON
        PORTB |= (1 << PINB5);        // Set PORTB5 On
        _delay_ms(300);
    }
}

else if(MCUSR & 4){
    MCUSR=0;
    for (i=0;i<8;i++)    {

        //LED OFF

```



```

        PORTB &= ~(1 << PINB5);      // Set PORTB5 Lo w
        _delay_ms(100);
        //LED ON
        PORTB |= (1 << PINB5);      // Set PORTB5 On
        _delay_ms(100);
    } }

else if(MCUSR & 8) {
    MCUSR=0;    WDT_Init();
for (i=0;i<8;i++)

    {        wdt_reset();

        //LED OFF
        PORTB &= ~(1 << PINB5);      // Set PORTB5 Lo w
        _delay_ms(20);
        //LED ON
        PORTB |= (1 << PINB5);      // Set PORTB5 On
        _delay_ms(80);
    }
    MCUSR=0;
}

//initialize watchdog
WDT_Init();

//delay to detect reset
//_delay_ms(500);
while(1){

```

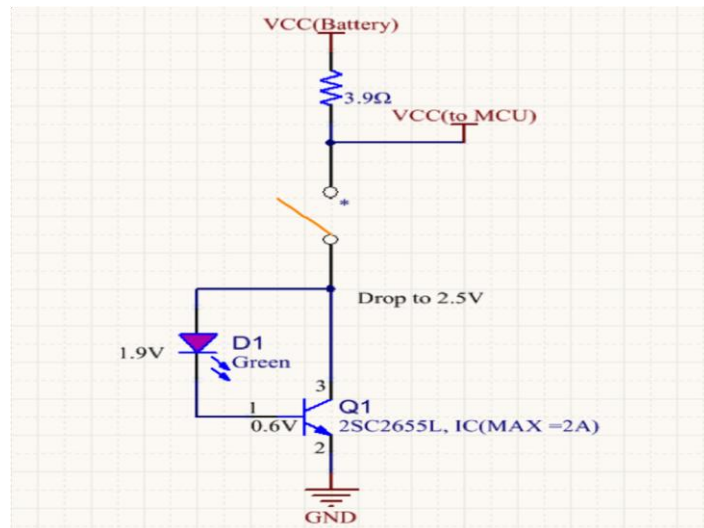
[illegible]

}

7

### Tarea 7 - Restablecer circuito de prueba

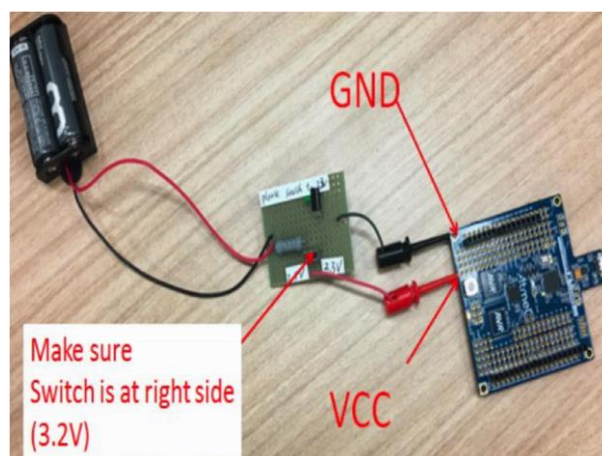
1. Cree el circuito de prueba de restablecimiento que se muestra en el esquema.



7

### Tarea 8 - Prueba POR

1. Extraiga el cable USB
2. Asegúrese de que el interruptor del cable VCC ajustable externo esté en el lado derecho (3.2 V)
3. Enganche el cable VCC ajustable externo a la placa



1. Resultado: El LED naranja responderá parpadeando (0,3 ms, cuatro veces) según la sección de código a continuación porque se detecta el restablecimiento de POR.

```
int main(void)
{
    unsigned char i;
    DDRB |= (1 << PINB5);           // Set PORTB5 as output ,
    DDRB &= ~(1<<PINB7);             //Set PORTB7 as input

    if(MCUSR & 1 ){                  //POR detected
        MCUSR=0;
        for ( i=0;i<4;i++)
        {
            //LED OFF
            PORTB &= ~(1 << PINB5); // Set PORTB5 Low
            _delay_ms(300);
            //LED ON
            PORTB |= (1 << PINB5);   // Set PORTB5 On
            _delay_ms(300);
        }
    }
    else if(MCUSR & 4) {
        MCUSR=0;
    }
}
```

9

### Tarea 9 - Prueba BOR

1. Después de la prueba POR, deslice el interruptor del cable VCC ajustable externo hacia el lado izquierdo (2,5 V)
2. Deslice el interruptor del cable VCC ajustable externo hacia el lado derecho (3,2 V)
3. Resultado: el LED naranja responderá parpadeando (0,1 ms, ocho veces) en función del código resaltado a continuación porque se detecta el restablecimiento de BOR.

```
else if(MCUSR & 4) {                //BOR reset detected
    MCUSR=0;
    for (i=0;i<8;i++)
    {
        //LED OFF
        PORTB &= ~(1 << PINB5);     // Set PORTB5 Low
        _delay_ms(100);
        //LED ON
        PORTB |= (1 << PINB5);      // Set PORTB5 On
        _delay_ms(100);
    }
}
else if(MCUSR & 8) {                // WDT reset detected
    MCUSR=0;
    WDT_Init();
    for (i=0;i<8;i++)
    {
        wdt_reset();
        //LED OFF
        PORTB &= ~(1 << PINB5);     // Set PORTB5 Low
        _delay_ms(20);
        //LED ON
        PORTB |= (1 << PINB5);      // Set PORTB5 On
        _delay_ms(80);
    }
}
```

10

### Tarea 10 - Prueba WDT

1. Presione el botón en la placa durante aproximadamente 1 segundo y luego suelte el botón.  
Los retrasos prolongados se activarán en la rutina principal, lo que provocará un tiempo de espera WDT.

Nota: El temporizador WDT está configurado en 2 segundos.

```
if((PINB&1<<PINB7)==0){  
    PORTB |= (1 << PINB5);           // Set PORTB5 high  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
    _delay_ms(500);  
}  
wdt_reset();  
}
```

1. Resultado: el LED naranja responderá parpadeando rápidamente (0,02 ms encendido, 0,08 ms apagado, cuatro veces) al principio cuando el WDT interrumpa isr:

```
//Watchdog timeout ISR  
ISR(WDT_vect)  
{  
    //Burst of 0.1Hz pulses  
    for (uint8_t i=0;i<4;i++)  
    {  
        //LED OFF  
        PORTB &= ~(1 << PINB5);           // Set PORTB5 Low  
        _delay_ms(80);  
        //LED ON  
        PORTB |= (1 << PINB5);           // Set PORTB5 On  
        _delay_ms(20);  
    }  
}
```

1. Resultado: el LED naranja volverá a responder parpadeando rápidamente (0,02 ms encendido, 0,08 ms apagado, cuatro veces) a medida que se detecte el restablecimiento de WDT:

```
else if(MCUSR & 8) {           // WDT reset detected  
    MCUSR=0;  
    WDT_Init();  
    for (i=0;i<8;i++)  
    {  
        wdt_reset();  
        //LED OFF  
        PORTB &= ~(1 << PINB5);           // Set PORTB5 Low  
        _delay_ms(20);  
        //LED ON  
        PORTB |= (1 << PINB5);           // Set PORTB5 On  
        _delay_ms(80);  
    }  
    MCUSR=0;  
}
```

//initialize watchdog

## Análisis

El proyecto muestra tres formas en que puede ocurrir un reinicio: POR, BOR y WDT Timeout. Cada uno tiene una aplicación única y todos pueden ejecutarse en la misma aplicación. Los ejemplos de código son solo una referencia a cómo se pueden configurar e implementar estos tipos de reinicios.

### **Conclusiones**

El proyecto ayuda a explicar cómo funciona el reinicio de circuitos dentro del AVR y cómo implementarlo. La sección de código se puede reutilizar en aplicaciones futuras que pueden requerir una estructura de restablecimiento similar. De ninguna manera es esta la única forma de diseñar reinicios en el dispositivo AVR, este es solo un proyecto de muestra simple que ayuda a explicar la operación y le permite aplicar su conocimiento y comprensión de la estructura de restablecimiento a una aplicación específica.