

# Descripción general del oscilador megaAVR®

Los microcontroladores de 8 bits megaAVR® de Microchip tienen varias opciones de fuente de reloj, seleccionables a través de la programación de los bits de fusible **CKSEL Flash** ([/8avr:avrfuses](#)) . Esta discusión es específica de la MCU ATmega328PB . (<http://www.microchip.com/wwwproducts/en/ATmega328PB>) Los bits de fusible pueden seleccionar uno de:

- Oscilador de cristal de baja potencia
- Oscilador de cristal de baja frecuencia
- Oscilador RC interno de 128 kHz
- Oscilador RC interno calibrado, y
- Reloj externo.



La **fuente del reloj del sistema** no se puede cambiar durante el tiempo de ejecución, ya que se configura a través de la programación de fusibles.



La **frecuencia del reloj del sistema** se puede cambiar durante el tiempo de ejecución escribiendo en el registro del preescalador del **reloj del sistema** ([/8avr:osc-mega-overview#system-clock-prescaler](#)) (CLKPR).

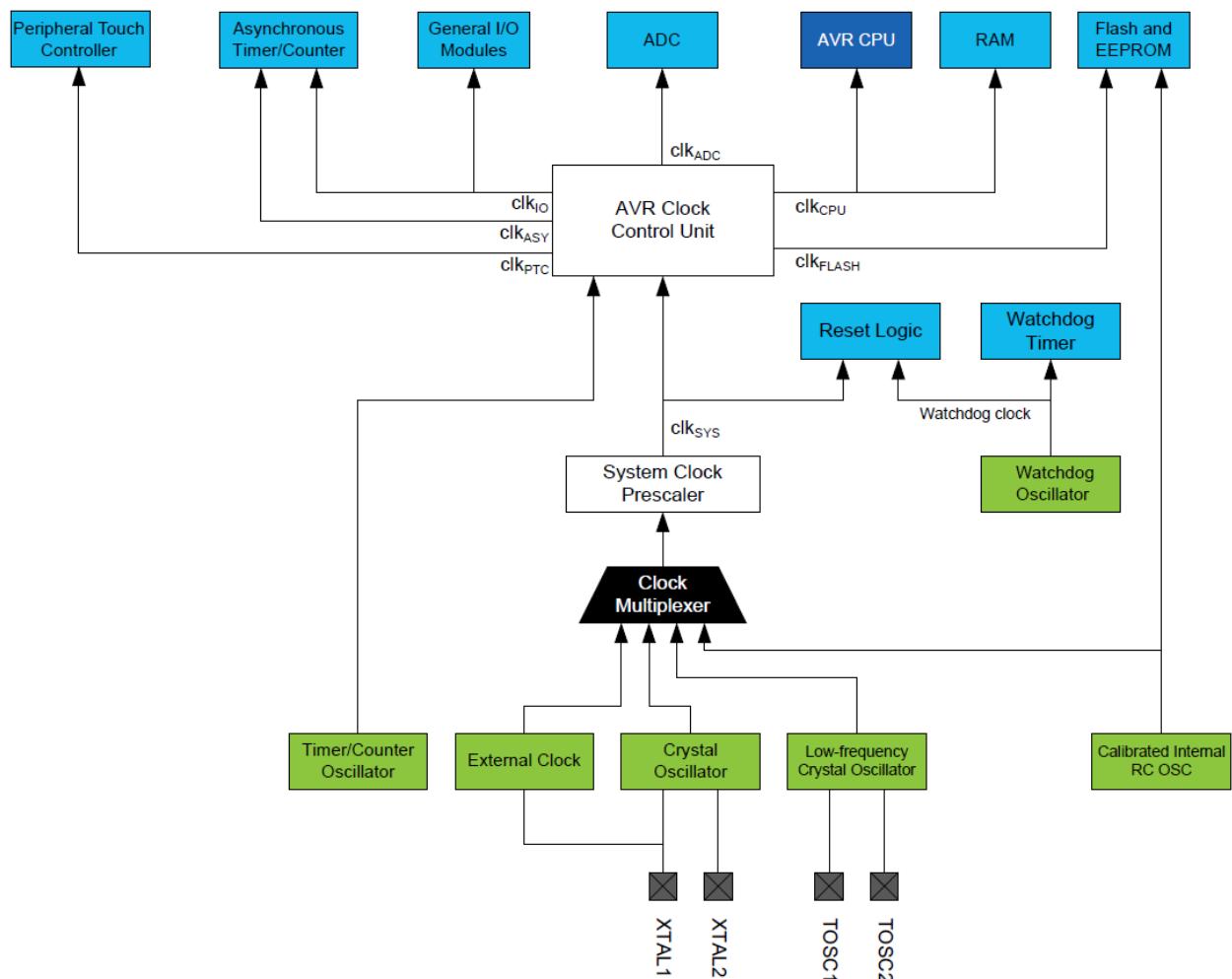
Cada fuente de reloj proporciona una opción de retraso después del reinicio o encendido del dispositivo para mantener el dispositivo reiniciado hasta que se suministre con Vcc mínimo. El reloj de la fuente seleccionada se ingresa al generador de reloj AVR® y se enruta a los módulos apropiados.



La **frecuencia operativa máxima del megaAVR® depende de V<sub>cc</sub>** . El software de la aplicación debe garantizar que la frecuencia de la fuente de reloj seleccionada se encuentre dentro del área de operación segura (consulte la sección 33.4 en la hoja de datos del dispositivo (<http://ww1.microchip.com/downloads/en/DeviceDoc/40001906A.pdf>)).

# Visión general

La siguiente figura ilustra los principales sistemas de reloj del dispositivo y su distribución. No es necesario que todos los relojes estén activos en un momento dado. Para reducir el consumo de energía, los relojes de los módulos que no se utilizan se pueden detener utilizando diferentes modos de suspensión (/8avr:avrsleep) . Los sistemas de reloj se describen en las siguientes secciones. La frecuencia del reloj del sistema se refiere a la frecuencia generada por el preescalador del reloj del sistema. Todas las salidas de reloj de la unidad de control de reloj AVR funcionan a la misma frecuencia.



(/local--files/8avr:osc-mega-overview/atmega328pb-sys-clk-distribution.png)

## Fuentes de reloj

El dispositivo tiene las siguientes opciones de fuente de reloj, seleccionables a través de los bits **CKSEL** Flash Fuse como se muestra a continuación. El reloj de la fuente seleccionada se ingresa al generador de reloj AVR® y se enruta a los módulos apropiados.

Device Clocking Option	CKSEL[3:0]
Low Power Crystal Oscillator	1111 - 1000
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

(/local--files/8avr:osc-mega-overview/atmega328pb-clk-sources.png)

## Fuente de reloj predeterminada

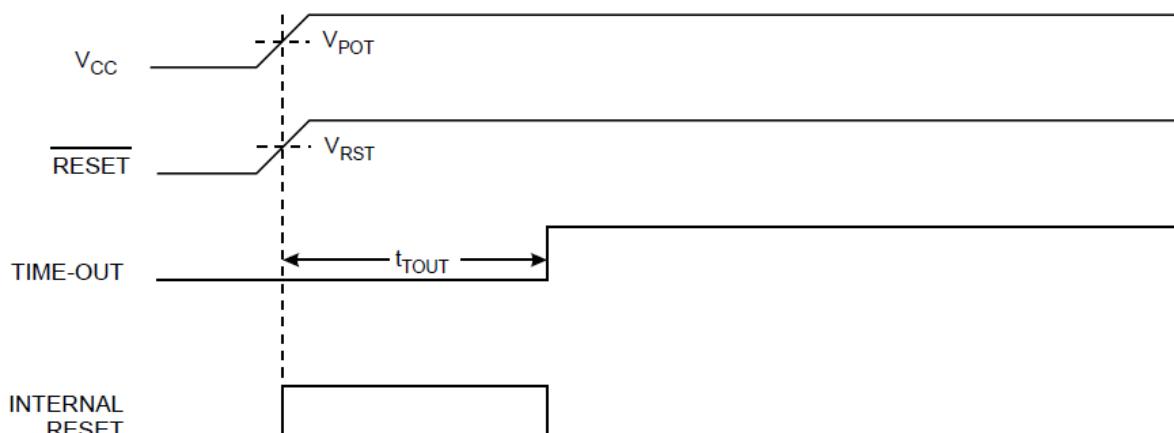
El dispositivo se envía con el oscilador RC interno seleccionado a 8,0 MHz y con el fusible CKDIV8 programado, lo que da como resultado un reloj del sistema de 1,0 MHz. El tiempo de inicio se establece al máximo y el período de tiempo de espera está habilitado: CKSEL=0010, SUT=10, CKDIV8=0. Esta configuración predeterminada garantiza que todos los usuarios puedan configurar la fuente de reloj deseada utilizando cualquier interfaz de programación disponible.

## Secuencia de inicio del reloj

Cualquier fuente de reloj necesita (i) un **V<sub>CC</sub> suficiente para comenzar a oscilar** y (ii) **un número mínimo de ciclos de oscilación antes de que pueda considerarse estable**.

### Estabilidad V<sub>CC</sub>

Para garantizar suficiente V<sub>CC</sub>, el dispositivo emite un restablecimiento interno con un retraso de tiempo de espera (  $t_{TOUT}$  ) después de que todas las demás fuentes de restablecimiento liberan el restablecimiento del dispositivo:



(/local--files/8avr:osc-mega-overview/atmega328pb-tout-delay.png)

El retardo (  $t_{TOUT}$  ) se cronometra desde el oscilador de vigilancia y el tiempo de retardo se establece mediante los bits de fusible SUTx y CKSELx. Los retardos seleccionables para  $t_{TOUT}$  se muestran en la siguiente tabla. Tenga en cuenta que la frecuencia del Watchdog Oscillator depende del voltaje:

Typ. Time-out ( $V_{CC} = 5.0V$ )	Typ. Time-out ( $V_{CC} = 3.0V$ )
0ms	0ms
4ms	4.3ms
65ms	69ms

(/local--files/8avr:osc-mega-overview/atmega328pb-tout-values.png)



$V_{CC}$  no se controla durante el retraso, por lo que se requiere seleccionar un retraso más largo que el tiempo de subida de  $V_{CC}$ . Si esto no es posible, se debe utilizar un circuito de detección de Brown-Out (BOD) interno o externo. Un circuito BOD garantizará suficiente  $V_{CC}$  antes de liberar el reinicio, y el retraso de tiempo de espera se puede desactivar. No se recomienda deshabilitar el retraso de tiempo de espera sin utilizar un circuito de detección de Brown-Out.

## Estabilidad del oscilador

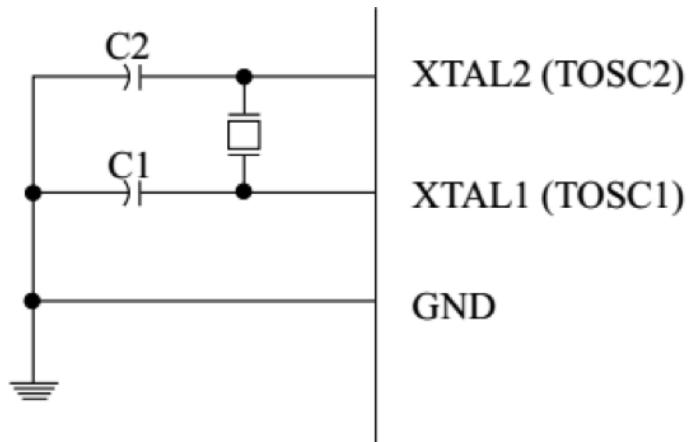
Se requiere que el oscilador oscile durante un número mínimo de ciclos antes de que el reloj se considere estable. Un contador de ondulación interno monitorea el reloj de salida del oscilador y mantiene activo el restablecimiento interno durante un número determinado de ciclos de reloj. Luego se libera el reinicio y el dispositivo comenzará a ejecutarse. El tiempo recomendado de puesta en marcha del oscilador depende del tipo de reloj y varía de 6 ciclos para un reloj aplicado externamente a 32K ciclos para un cristal de baja frecuencia.



Consulte la sección 11 en la hoja de datos del dispositivo (<http://ww1.microchip.com/downloads/en/DeviceDoc/40001906A.pdf>) que especifica el número de ciclos de retardo de CK para cada tipo de fuente de reloj y ajuste de fusible SUTx.

## Oscilador de cristal de baja potencia

Los pines XTAL1 y XTAL2 son entrada y salida, respectivamente, de un amplificador inversor que puede configurarse para usarse como un oscilador en chip, como se muestra en la figura a continuación. Puede usarse un cristal de cuarzo o un resonador cerámico:



(/local--files/8avr:osc-mega-overview/atmega328pb-xtal-connection.png)

El oscilador de baja potencia puede operar en tres modos diferentes, cada uno optimizado para un rango de frecuencia específico. El modo de operación es seleccionado por los fusibles CKSEL[3:1], como se muestra en la siguiente tabla:

Frequency Range [MHz]	CKSEL[3:1] <sup>(2)</sup>	Range for total capacitance of C1 and C2 [pF] <sup>(4)</sup>
0.4 - 0.9	100 <sup>(3)</sup>	–
0.9 - 3.0	101	12 - 22
3.0 - 8.0	110	12 - 22
8.0 - 16.0	111	12 - 22

(/local--files/8avr:osc-mega-overview/atmega328pb-xtal-modes.png)

El Fusible CKSEL0 junto con los Fusibles SUT[1:0] seleccionan los tiempos de arranque (ver sección 11.3 en la hoja de datos del dispositivo (<http://ww1.microchip.com/downloads/en/DeviceDoc/40001906A.pdf>)).

## Oscilador de cristal de baja frecuencia

El oscilador de cristal de baja frecuencia está optimizado para su uso con un cristal de reloj de 32,768 kHz. El oscilador de cristal de baja frecuencia debe seleccionarse configurando los fusibles CKSEL en '0110' o '0111', y los tiempos de inicio están determinados por el SUT fusibles

## Oscilador RC interno calibrado

De forma predeterminada, el oscilador RC interno proporciona un reloj de 8,0 MHz. Aunque depende del voltaje y la temperatura, el usuario puede calibrar este reloj con mucha precisión. El dispositivo se envía con el fusible CKDIV8 programado, lo que proporciona una frecuencia de reloj del sistema de 1 MHz. Este reloj se puede seleccionar como el reloj del sistema programando los fusibles CKSEL en '0010'. Si se

selecciona, funcionará sin componentes externos. Durante el reinicio, el hardware carga el valor de calibración preprogramado en el registro OSCCAL y, por lo tanto, calibra automáticamente el oscilador RC.



Consulte la nota de aplicación AVR053 ([http://ww1.microchip.com/downloads/en/AppNotes/Atmel-2555-Internal-RC-Oscillator-Calibration-for-tinyAVR-and-megaAVR-Devices\\_ApplicationNote\\_AVR053.pdf](http://ww1.microchip.com/downloads/en/AppNotes/Atmel-2555-Internal-RC-Oscillator-Calibration-for-tinyAVR-and-megaAVR-Devices_ApplicationNote_AVR053.pdf)) que describe el procedimiento para volver a calibrar el oscilador RC interno.

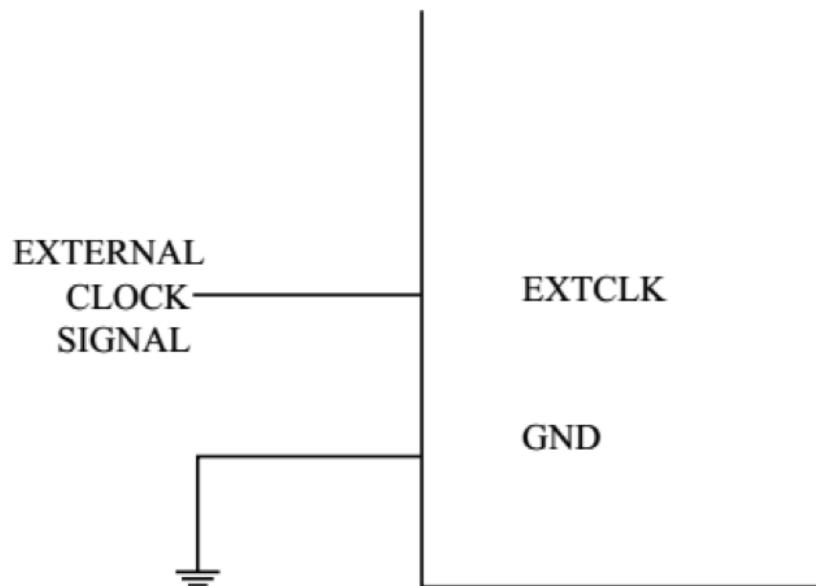
## Oscilador interno de 128 kHz

El oscilador interno de 128 kHz es un oscilador de baja potencia que proporciona un reloj de 128 kHz. Este reloj se puede seleccionar como reloj del sistema programando los fusibles CKSEL en '0011'.

## Reloj externo

Para controlar el dispositivo desde una fuente de reloj externa, EXTCLK debe controlarse como se muestra en la figura a continuación. Para ejecutar el dispositivo en un reloj externo, los fusibles CKSEL deben programarse en '0000'.

### External Clock Drive Configuration



(/local--files/8avr:osc-mega-overview/atmega328pb-ext-clk-connection.png)

## Búfer de salida de reloj

El dispositivo puede generar el reloj del sistema en el pin CLKO. Para habilitar la salida, se debe programar el Fusible CKOUT. Este modo es adecuado cuando el reloj del chip

se usa para controlar otros circuitos en el sistema. El reloj también se emitirá durante el

reinicio, y el funcionamiento normal del pin de E/S se anulará cuando se programe el fusible. Cualquier fuente de reloj, incluido el oscilador RC interno, se puede seleccionar cuando el reloj se emite en CLKO. Si se utiliza el preescalador de reloj del sistema, lo que se emite es el reloj del sistema dividido.

## Temporizador/Contador Oscilador

El dispositivo utiliza el mismo oscilador de cristal para el oscilador de baja frecuencia y el temporizador/contador de osciladores. Consulte Oscilador de cristal de baja frecuencia para obtener detalles sobre los requisitos del oscilador y del cristal.

En este dispositivo, los pines del oscilador del temporizador/contador (TOSC1 y TOSC2) se comparten con EXTCLK. Cuando se utiliza el temporizador/contador de osciladores, el reloj del sistema debe ser cuatro veces la frecuencia del oscilador. Debido a esto y al uso compartido de pines, el temporizador/contador de osciladores solo se puede usar cuando el oscilador RC interno calibrado se selecciona como fuente de reloj del sistema. Se puede aplicar una fuente de reloj externa a TOSC1 si el bit Habilitar entrada de reloj externo en el registro de estado asíncrono (ASSR.EXCLK) se escribe en '1'. Consulte la descripción de la operación asíncrona del temporizador/contador2 para obtener una descripción más detallada sobre la selección de un reloj externo como entrada en lugar de un cristal de reloj de 32,768 kHz.

## Prescaler del reloj del sistema

El dispositivo tiene un preescalador de reloj del sistema, y el reloj del sistema se puede dividir configurando el Registro de preescala de reloj (CLKPR). Esta función se puede utilizar para disminuir la frecuencia del reloj del sistema y el consumo de energía cuando el requisito de potencia de procesamiento es bajo. Esto se puede usar con todas las opciones de fuente de reloj y afectará la frecuencia de reloj de la CPU y todos los periféricos síncronos. clk<sub>I/O</sub>, clk<sub>ADC</sub>, clk<sub>CPU</sub> y clk<sub>FLASH</sub> se dividen por un factor como se muestra en la descripción de CLKPR:

**Name:** CLKPR  
**Offset:** 0x61  
**Reset:** Refer to the bit description  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	CLKPCE				CLKPSn	CLKPSn	CLKPSn	CLKPSn
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				x	x	x	x

#### Bit 7 – CLKPCE: Clock Prescaler Change Enable

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

#### Bits 3:0 – CLKPSn: Clock Prescaler Select n [n = 3:0]

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in the table below.

(/local--files/8avr:osc-mega-overview/atmega328pb-clkpr-1.png)

CLKPS[3:0]	Clock Division Factor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

(/local--files/8avr:osc-mega-overview/atmega328pb-clkpr-2.png)

## Escribiendo a CLKPR

Al cambiar entre las configuraciones del preescalador, el Preescalador del reloj del sistema asegura que no ocurran fallas en el sistema del reloj. También asegura que ninguna frecuencia intermedia sea superior a la frecuencia de reloj correspondiente a la

configuración anterior, ni a la frecuencia de reloj correspondiente a la nueva

configuración. El contador de ondas que implementa el preescalador se ejecuta a la frecuencia del reloj indiviso, que puede ser más rápido que la frecuencia del reloj de la CPU. Por lo tanto, no es posible determinar el estado del preescalador; incluso si fuera legible, el tiempo exacto que se tarda en cambiar de una división de reloj a otra no se puede predecir con exactitud. Desde el momento en que se escriben los valores de los bits de selección del preescalador de reloj (CLKPS[3:0]), transcurren entre  $T_1 + T_2$  y  $T_1 + 2 * T_2$  antes de que se active la nueva frecuencia de reloj. En este intervalo, se producen dos flancos de reloj activos. Aquí,  $T_1$  es el período de reloj anterior y  $T_2$  es el período correspondiente a la nueva configuración del preescalador. Para evitar cambios involuntarios de la frecuencia del reloj, se debe seguir un procedimiento de escritura especial para cambiar los bits CLKPS:

1. Escriba el bit de habilitación de cambio de preescalador de reloj (CLKPCE) en '1' y todos los demás bits en CLKPR en cero: CLKPR=0x80.
2. Dentro de cuatro ciclos, escriba el valor deseado en CLKPS[3:0] mientras escribe un cero en CLKPCE: CLKPR=0x0N



Las interrupciones deben desactivarse al cambiar la configuración del preescalador para asegurarse de que el procedimiento de escritura no se interrumpa.

## Ejemplo de código

La siguiente función se puede utilizar para actualizar dinámicamente CLKPR como se requiere anteriormente. Tenga en cuenta el uso de las funciones `cli()` y `sei()` para garantizar que el procedimiento de escritura CLKPR no se interrumpa.



```

1  #include <stdint.h> // St...
2  #include <avr/io.h> // SFR
3  #include <avr/interrupt.h> /
4
5  void clkPrescaleSet(uint8_t
6      cli();
7      CLKPR = (1 << CLKPCE);
8      CLKPR = divisionFactor;
9      sei();
10 }

```



Para ver esta función en uso, visite el proyecto de ejemplo del **oscilador megaAVR® (/8avr:osc-mega-example)**

## CLKDIV8 Fusible y CLKPR

El fusible CKDIV8 determina el valor inicial de los bits CLKPS. Si CKDIV8 no está programado, los bits CLKPS se restablecerán a "0000". Si se programa CKDIV8, los

programado, los bits CLKPS se restablecerán a "0000". Si se programa CKDIV8, los bits CLKPS se restablecen a "0011", dando un factor de división de 8 al inicio. Esta función debe utilizarse si la fuente de reloj seleccionada tiene una frecuencia superior a la frecuencia máxima del dispositivo en las condiciones de funcionamiento actuales. Tenga en cuenta que se puede escribir cualquier valor en los bits CLKPS independientemente de la configuración del fusible CKDIV8. El software de la aplicación debe garantizar que se elija un factor de división suficiente si la fuente de reloj seleccionada tiene una frecuencia superior a la frecuencia máxima del dispositivo en las condiciones de funcionamiento actuales. El dispositivo se envía con el fusible CKDIV8 programado.

## Aprende más



### Proyecto de ejemplo de oscilador megaAVR®

Más información > (/8avr:osc-mega-example)

# Descripción general de AVR® USART

Los microcontroladores Microchip AVR® de 8 bits contienen un periférico de comunicación altamente flexible conocido como **USART** (receptor y transmisor serie universal síncrono y asíncrono). Este periférico se puede utilizar para comunicarse con una amplia variedad de otros componentes, incluidos otros microcontroladores, módulos inalámbricos, pantallas LCD, módulos GPS, etc. El periférico USART puede funcionar en uno de dos modos principales: sincrónico o asincrónico. Este módulo se centra en el modo de funcionamiento **asíncrono** ([https://en.wikipedia.org/wiki/Universal\\_asynchronous\\_receiver/transmitter](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter)).

([https://en.wikipedia.org/wiki/Universal\\_asynchronous\\_receiver/transmitter](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter))

## Aprende más



### Configuración megaAVR® USART

Más información > (/8avr:uart-mega-configuration)



### MegaAVR® USART Ejemplo (sondeo)

Más información > (/8avr:uart-mega-example-polled)



# Configuración megaAVR® USART

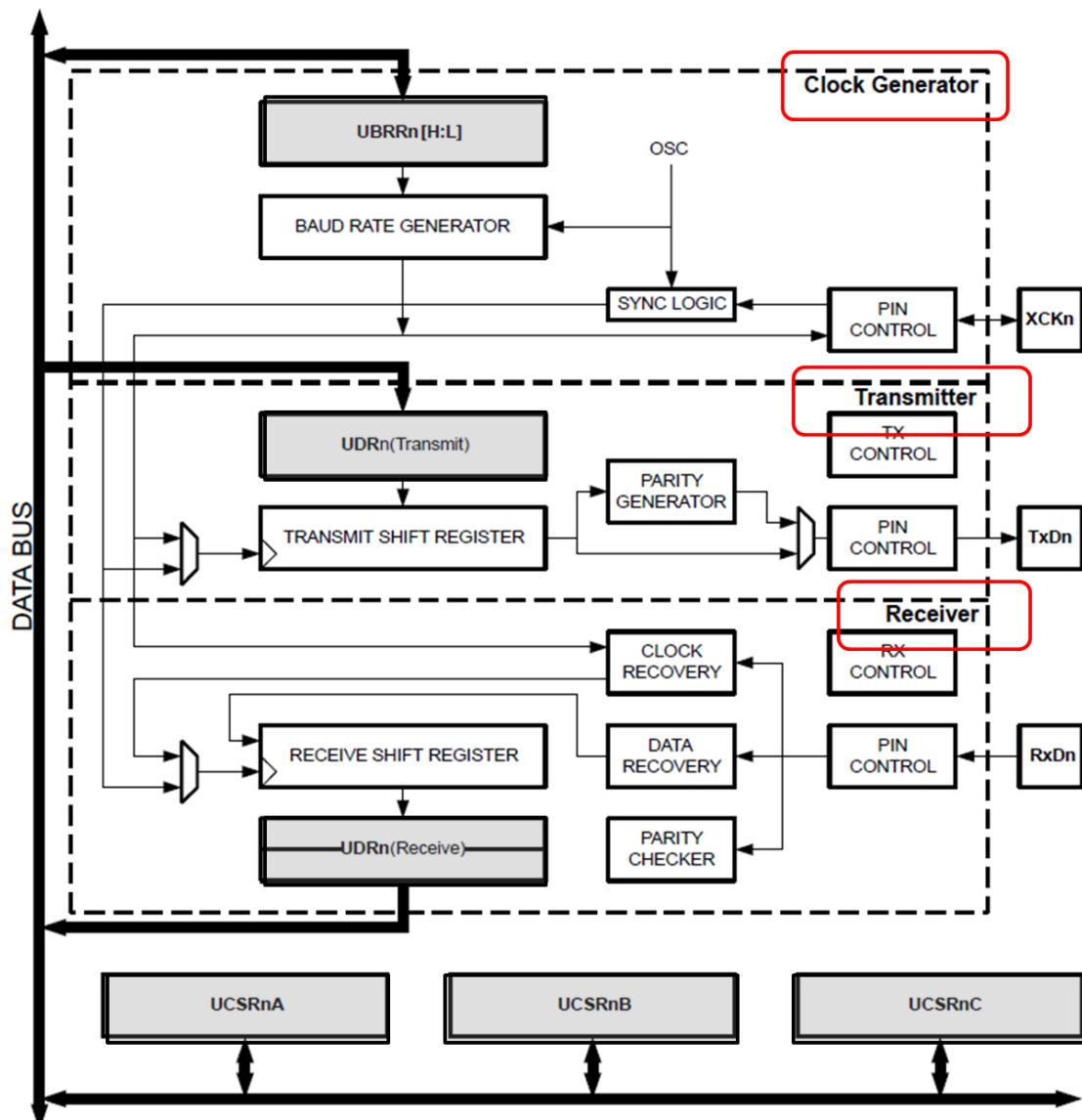
---

En esta sección, cubriremos los pasos básicos de codificación necesarios para configurar/usar el módulo USART en un MCU megaAVR®, con un enfoque en el dispositivo **ATmega328PB** (<http://www.microchip.com/wwwproducts/en/ATmega328PB>)

## Visión general

El módulo USART consta de tres secciones principales, como se muestra en el siguiente diagrama: **generador de reloj** , **transmisor** y **receptor** .

Figure 24-1. USART Block Diagram



(/local--files/8avr:uart-mega-configuration/uart-block-diagram.png)



Los registros clave (resaltados en gris) incluyen:

- Registros de control y estado ( **UCSRnA** , **UCSRnB** , **UCSRnC** ) compartidos por las tres secciones.
- Registro de datos **UDRn** compartido por las secciones Transmisor y Receptor.
- Registros de control de velocidad en baudios **UBRRn[H:L]** utilizados por el generador de reloj.



" n " en el nombre del registro/bit identifica la instancia de hardware USART específica (0, 1, 2) a la que está asociado el registro/bit. Por ejemplo , **UCSR0A** se refiere a **USART0** Control & Status Register A

## Usando el USART (Resumen)

Para la operación sondeada básica, se deben realizar los siguientes pasos mínimos:

1. Elija una tasa de baudios y programe los **registros UBRRn[H:L]** en **consecuencia**

CONSIDERACIONES.

2. Habilite las secciones de transmisión y recepción en serie de USART.
3. Si está transmitiendo, espere hasta que el registro de desplazamiento de transmisión esté vacío (sondee en **UCSRnA.UDREn** ), luego cargue su byte de datos en **UDRn** .
4. Si recibe, espere hasta que se establezca el bit de recepción de datos del receptor (sondee en **UCSRnA.RXCn** ), luego lea los datos de **UDRn** . La lectura de UDRn borra automáticamente el bit y prepara el hardware para recibir el siguiente byte.

## Inicialización

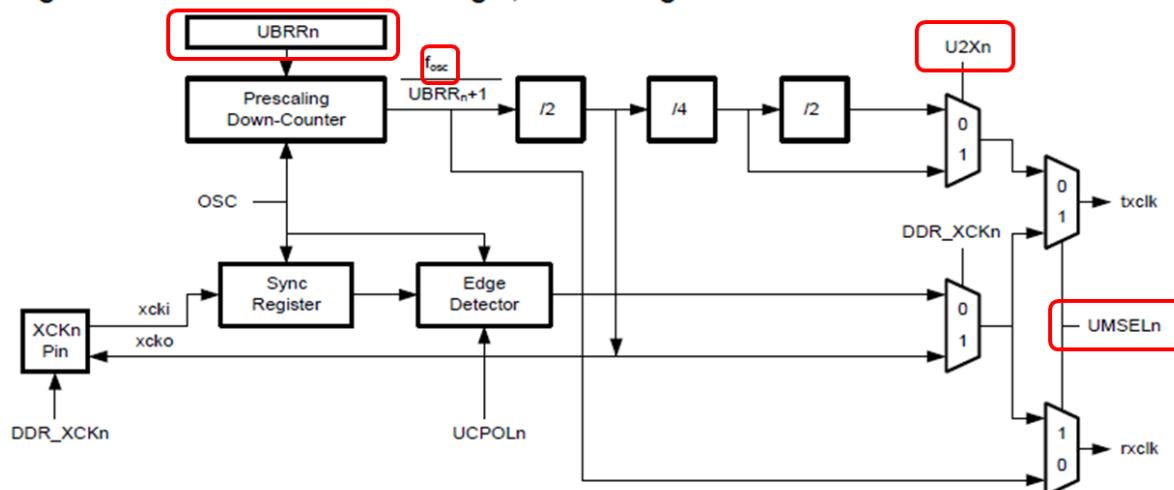
El USART debe inicializarse antes de que pueda tener lugar cualquier comunicación. El proceso de inicialización normalmente consiste en:

- Configuración de la velocidad en baudios,
- Configuración del formato de cuadro y
- Habilitación del Transmisor o del Receptor según el uso.

## Configuración de la tasa de baudios

La generación de reloj interno se utiliza para el modo de funcionamiento asíncrono. La lógica de generación de reloj genera el reloj base para el transmisor y el receptor (los registros clave y los bits de control están resaltados):

Figure 24-2. Clock Generation Logic, Block Diagram



Signal description:

- txclk: Transmitter clock (internal signal).
- rxclk: Receiver base clock (internal signal).
- xcki: Input from XCKn pin (internal signal). Used for synchronous slave operation.
- xcko: Clock output to XCKn pin (internal signal). Used for synchronous master operation.
- fosc: System clock frequency.

(/local--files/8avr:usart-mega-configuration/usart-clock-generator-diagram.png)

## Selección de modo USART (UMSEL<sub>n</sub>)

La ecuación de velocidad en baudios utilizada por el módulo se establece en función del modo de funcionamiento. Para la operación en modo asíncrono, los bits de selección de modo USART en el registro C de control y estado de USART ( **UCSRnC.UMSEL<sub>n</sub>[1:0]** ) se utilizan para seleccionar **la operación asíncrona (UMSEL[1:0] = 00)** como se muestra:

Name: UCSR0C, UCSR1C  
 Offset: 0xC2 + n\*0x08 [n=0..1]  
 Reset: 0x06  
 Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

(/local--files/8avr:uart-mega-configuration/uart-mode-select-bits.png)

## Modo de doble velocidad (U2X<sub>n</sub>)

Para el modo asíncrono, la tasa de TX de USART se puede duplicar configurando el bit U2X<sub>n</sub> en el registro UCSRnA ( **UCSRnA.U2X<sub>n</sub> = 1** ).



Con el modo de velocidad doble configurado, el receptor solo usará la mitad del número de muestras (reducidas de 16 a 8) para el muestreo de datos y la recuperación del reloj y, por lo tanto, se requiere una configuración de velocidad en baudios y un reloj del sistema más precisos cuando se usa este modo.

## Registro de velocidad en baudios (UBRR<sub>n</sub>)

El registro de tasa de baudios USART ( **UBRR<sub>n</sub>** ) y el contador descendente conectado a él funcionan como un preescalador programable o generador de tasa de baudios. El contador regresivo, que se ejecuta en el reloj del sistema ( $f_{osc}$ ), se carga con el valor UBRR<sub>n</sub> cada vez que el contador llega a cero o cuando se escribe el registro UBRRnL. Se genera un reloj cada vez que el contador llega a cero. Este reloj es la salida del reloj del generador de velocidad en baudios ( $= f_{osc}/(UBRRn+1)$ ). El transmisor divide la salida del reloj del generador de velocidad en baudios por 2, 8 o 16 según el modo. La salida del generador de velocidad en baudios es utilizada directamente por el reloj del receptor y las unidades de recuperación de datos. Sin embargo, las unidades de recuperación usan una máquina de estado que usa 2, 8 o 16 estados según el modo establecido por el estado de los bits UMSel, U2X<sub>n</sub> y DDR\_XCK. La siguiente tabla contiene ecuaciones para calcular la tasa de baudios (en bits por segundo) y para calcular el valor UBRR<sub>n</sub> para cada modo de operación utilizando una fuente de reloj generada internamente.

Table 24-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{osc}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$

(/local--files/8avr:uart-mega-configuration/uart-baud-equations.png)

- **BAUDIOS** : Tasa de baudios (en bits por segundo, bps)
- **f<sub>osc</sub>** : Frecuencia de reloj del oscilador del sistema
- **UBRRn** : Contenido de los Registros UBRRnH y UBRRnL, (0-4095).



La biblioteca **AVR-LIBC Setbaud** ([http://www.nongnu.org/avr-libc/user-manual/group\\_\\_util\\_\\_setbaud.html](http://www.nongnu.org/avr-libc/user-manual/group__util__setbaud.html)) contiene macros útiles para calcular los valores correctos para escribir en los registros UBRRnH y UBRRnL. Consulte el ejemplo de código de inicialización a continuación.

También se proporcionan tablas en la hoja de datos del dispositivo que contienen valores UBRRn para tasas de baudios comunes, dadas varias frecuencias de oscilador:

Table 24-9. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f <sub>osc</sub> = 16.0000MHz				f <sub>osc</sub> = 18.4320MHz				f <sub>osc</sub> = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%

(/local--files/8avr:uart-mega-configuration/uart-common-ubrrn-values.png)



Para los cálculos de frecuencia en baudios, generalmente se acepta que los porcentajes de error de menos de  $\pm 2\%$  son aceptables.

## Configuración del formato de marco

El registro C de control y estado de USART ( **UCSRnC** ) se utiliza para configurar el formato de la trama de comunicación UART: paridad, número de bits de parada y número de bits de datos. Los ajustes para el formato de cuadro típico “8N1” son los siguientes:

- **UPM[1:0] = 00** para Sin paridad
- **USBS = 0** para 1 bit de parada
- **UCSZ1[1:0] = 11** para 8 Bits

Name: UCSR0C, UCSR1C  
 Offset: 0xC2 + n\*0x08 [n=0..1]  
 Reset: 0x06  
 Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Reset values: Bit 7: 0, Bit 6: 0, Bit 5: 0, Bit 4: 0, Bit 3: 0, Bit 2: 1, Bit 1: 1, Bit 0: 0

(/local--files/8avr:uart-mega-configuration/uart-frame-format-settings.png)

## Habilitación del transmisor

El transmisor USART se habilita configurando el bit de **habilitación de transmisión (TXEN)** en el registro **UCSRnB** :

Name: UCSR0B, UCSR1B  
 Offset: 0xC1 + n\*0x08 [n=0..1]  
 Reset: 0x00  
 Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

Reset values: Bit 7: 0, Bit 6: 0, Bit 5: 0, Bit 4: 0, Bit 3: 0, Bit 2: 0, Bit 1: 0, Bit 0: 0

(/local--files/8avr:uart-mega-configuration/uart-ucsrnb-txen.png)

Cuando el transmisor está habilitado, la operación normal del puerto del pin TxDn es anulada por el USART y se le asigna la función de salida en serie del transmisor.



La velocidad en baudios, el modo de operación y el formato de trama deben configurarse una vez antes de realizar cualquier transmisión.

## Habilitación del receptor

El receptor USART se habilita escribiendo el bit de **habilitación de recepción (RXEN)** en el registro **UCSRnB** a '1':

**Name:** UCSR0B, UCSR1B  
**Offset:** 0xC1 + n\*0x08 [n=0..1]  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

(/local--files/8avr:uart-mega-configuration/uart-ucsrnb-rxen.png)

Cuando el Receptor está habilitado, el USART anula la operación normal del puerto del pin RxDn y se le asigna la función como entrada en serie del Receptor.



La velocidad en baudios, el modo de operación y el formato de trama deben configurarse una vez antes de realizar cualquier transmisión.

## Ejemplo de código

El siguiente ejemplo de código de inicialización de USART utiliza la biblioteca de utilidades **setbaud** ([http://www.nongnu.org/avr-libc/user-manual/group\\_\\_util\\_\\_setbaud.html](http://www.nongnu.org/avr-libc/user-manual/group__util__setbaud.html)) en AVR-LIBC. Esta biblioteca proporciona macros que usan el preprocesador c para calcular los valores apropiados para **UBBRn**. **Entradas** Este archivo de encabezado requiere que los valores de entrada ya estén definidos para **F\_CPU** y **BAUD**. Además, la macro **BAUD\_TOL** definirá la tolerancia de velocidad en baudios (en porcentaje) que es aceptable durante los cálculos. El valor de **BAUD\_TOL** por defecto será +/- 2%. **Salidas** Suponiendo que los **BAUD** solicitados son válidos para la **F\_CPU** dada, entonces la macro **UBRR\_VALUE**

se establece en el valor del preescalador requerido. Se proporcionan dos macros adicionales para los bytes alto y bajo del preescalador, respectivamente: **UBRRL\_VALUE** se establece en el byte inferior de **UBRR\_VALUE** y **UBRRH\_VALUE** se establece en el byte superior. Se definirá una macro adicional **USE\_2X**. Su valor se establece en 1 si la tasa de BAUDIOS deseada dentro de la tolerancia dada solo se puede lograr al establecer el bit **U2Xn** en la configuración de UART. Se definirá a 0 si no se necesita **U2Xn**.



```
1 #define F_CPU 16000000UL ? ▲
2 #define BAUD 38400UL
3 #define BAUD_TOL 2
4
5 #include <avr/io.h>
6 #include <util/setbaud.h>
7
8 void USART0_Init(void){
9
10    // Set the BAUD rate
11
12    UBRR0H = UBRRH_VALUE;
13    UBRR0L = UBRLL_VALUE;
14    #if USE_2X
15    UCSR0A |= (1 << U2X0);
16    #else
17    UCSR0A &= ~(1 << U2X0);
18    #endif
19
20    // Set the Mode & Frame
21
22    UCSR0C = 0x06;
23
24    // Enable USART0 Transmi
25
26    UCSR0B = (1 << TXEN0) | ▼
27
28 }
```



La biblioteca setbaud genera mensajes de advertencia durante la compilación si los parámetros de entrada generan una configuración de tasa BAUD que producirá una tasa de baudios fuera del BAUD\_TOL deseado.

## Transmisión de datos

### Transmitir

Una transmisión de datos se inicia cargando el búfer de transmisión con los datos a transmitir. La CPU puede cargar el búfer de transmisión escribiendo en el registro **UDRn** . Para la operación de sondeo, el firmware debe monitorear el indicador de registro de datos vacío ( **UCSRnA.UDREn** ) antes de cargar **UDRn** . Los datos almacenados en el búfer de transmisión se moverán al registro de desplazamiento cuando el registro de desplazamiento esté listo para enviar una nueva trama. El registro de desplazamiento se carga con nuevos datos si está en estado inactivo (sin transmisión en curso) o inmediatamente después de que se transmita el último bit de parada de la trama anterior. Cuando el registro de desplazamiento se carga con nuevos datos, transferirá un cuadro completo a la velocidad dada por el registro de baudios.

**Name:** UDR  
**Offset:** 0xC6 + n\*0x08 [n=0..1]  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
TXB / RXB[7:0]								
Access	R/W							
Reset	0	0	0	0	0	0	0	0

(/local--files/8avr:usart-mega-configuration/usart-udrn.png)



El indicador de interrupción de transmisión completa ( **USCRnA.TXCn** ) se establece y se puede generar una interrupción de TX opcional (si está habilitada) cuando se ha desplazado todo el marco en el registro de desplazamiento. El bit indicador **USCRnA.TXCn** se borra automáticamente cuando se ejecuta una interrupción de transmisión completa, o se puede borrar escribiendo un uno en su ubicación de bit.

## Recibir

El receptor inicia la recepción de datos cuando detecta un bit de inicio válido. Cada bit que sigue al bit de inicio se muestreará a la velocidad en baudios o al reloj XCKn, y se desplazará al registro de desplazamiento de recepción hasta que se reciba el primer bit de parada de una trama. El búfer de recepción se puede leer leyendo el registro **UDRn** . La recepción completa de un byte se puede verificar sondeando el bit RXCn en el registro **USCRnA** .



El indicador de interrupción de recepción completa ( **RXCn** ) se establece y se puede generar una interrupción de RX opcional (si está habilitada) cuando el cuadro completo en el registro de desplazamiento se ha copiado en el registro **UDRn** . Esta es una interrupción *persistente* , es decir, el firmware debe leer los datos recibidos de **UDRn** para borrar el indicador **RXCn**

## Ejemplo de código

Las siguientes API de bloqueo simple envían y reciben un byte de datos a través de USART0.



```
1 void USART0_Transmit(unsigned char data) {
2
3     // Wait for empty transmit buffer
4     while(!(UCSR0A & (1 << UDRE0)))
5
6     // Put data into buffer, and trigger the interrupt
7     UDR0 = data;
8
9 }
10
11 unsigned char USART0_Receive() {
12
13     // Wait for data to be received
14     while(!(UCSR0A & (1 << RXC0)))
15
16     // Get and return received character
17     return UDR0;
18
19 }
```

## Aprende más



### Configuración de interrupciones megaAVR®

Más información > ([/8avr:interrupts-mega-configuration](#))



### MegaAVR® USART Ejemplo (sondeo)

Más información > ([/8avr:uart-mega-example-polled](#))

# Configuración de interrupciones megaAVR®

El desarrollador de la aplicación debe inicializar cuidadosamente la operación de interrupción de AVR®. Esta página resume los pasos clave de inicialización y uso necesarios para usar interrupciones en una aplicación. Se proporciona más información sobre el uso de interrupciones en la sección **Módulo de interrupción de la biblioteca AVR-LIBC** ([http://www.nongnu.org/avr-libc/user-manual/group\\_\\_avr\\_\\_interrupts.html](http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html)) AVR-LIBC (<http://www.nongnu.org/avr-libc/>) .

([http://www.nongnu.org/avr-libc/user-manual/group\\_\\_avr\\_\\_interrupts.html](http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html))  
(<http://www.nongnu.org/avr-libc/>)

## Paso 1. #incluye encabezados estándar

La aplicación debe incluir archivos de encabezado `avr/io.h` y `avr/interrupt.h` como se muestra a continuación:



A screenshot of a code editor showing a C file. The file contains the following code:

```
1 #include <avr/io.h> ?
2 #include <avr/interrupt.h>
```

The code editor has a file icon with a '.c' extension in the top-left corner. The code is displayed in a text area with syntax highlighting. There are navigation arrows at the bottom of the code area.

El archivo de encabezado `avr/interrupt.h` proporciona varias macros destinadas a simplificar la aplicación de interrupciones en una aplicación, como macros para habilitar/deshabilitar interrupciones globalmente (bit 1 en el registro de estado), así como una macro para asignar una interrupción. función a un vector de interrupción específico:

- `si( )`
- `CLI( )`
- `ISR(vector_id, atributos)`

Las macros `vector_id` se definen en el archivo de encabezado específico del procesador (incluido a través de `avr/io.h` ), así como en la hoja de datos del dispositivo. Su construcción se define a continuación.

## Paso 2. Proporcionar rutina de servicio de interrupción

Una función de manejo de interrupciones es diferente a una función ordinaria en que maneja el contexto guardar y restaurar para asegurar que al regresar de la interrupción, se mantenga el contexto del programa. También se usa una secuencia de código diferente para regresar de estas funciones.

Hay varias acciones que el compilador debe realizar para generar una rutina de servicio de interrupción:

- Se debe indicar al compilador que use una forma alternativa de instrucción de retorno ( `RETI` vs. `RET` )
- Se debe informar al compilador sobre cualquier opción adicional específica
  - Habilitar el anidamiento de interrupciones
  - Opciones para la generación de código de prólogo/epílogo
- La función debe vincularse a un vector de interrupción específico.

Se proporcionan varios atributos de función de controlador al desarrollador de la aplicación, lo que habilita estas opciones.

- La macro `ISR( )` se proporciona para facilitar la definición de funciones de manejo de interrupciones con atributos



Para todos los vectores de interrupción sin controladores específicos, se instalará un controlador de interrupción predeterminado: **el controlador de interrupción predeterminado restablecerá el dispositivo**. Una aplicación puede anular el controlador predeterminado y proporcionar un controlador de interrupción predeterminado específico de la aplicación utilizando **BADISR\_vect** `vector_id` dentro de la macro `ISR( )`.

### Macro `ISR( )`

El siguiente ejemplo de código muestra cómo usar la macro `ISR()` para definir una función de interrupción:



```
1 ISR(vector_id, ISR_[BLOCK|NOBLOCK])
2 {
3     /* Hardware auto-clears the interrupt cause */
4     /* Clear the cause of the interrupt */
5     /* ISR-specific processing */
6 }
```

Los diversos parámetros se describirán ahora con más detalle.

## id\_vector

Este identificador es una *concatenación* de un **Vector Source ID** y **\_vect**. Los ID de fuente de vector se encuentran en la hoja de datos del dispositivo, como se muestra (parcialmente) en el siguiente ejemplo para ATmega328PB:

### 16.1. Interrupt Vectors in ATmega328PB

Table 16-1 Reset and Interrupt Vectors in ATmega328PB

Vector No	Program Address	Source	Interrupts definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
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	SPI0_STC	SPI1 Serial Transfer Complete
19	0x0024	USART0_RX	USART0 Rx Complete
20	0x0026	USART0_UDRE	USART0, Data Register Empty
21	0x0028	USART0_TX	USART0, Tx Complete
22	0x002A	ADC	ADC Conversion Complete

(/local--files/8avr:interrupts-mega-configuration/vector-source-id-328pb.png)



Los vector\_ids mal escritos **aún generarán una función**, sin embargo, no se conectará a la tabla de vectores de interrupción. El compilador generará una advertencia si detecta un nombre sospechoso.

## Atributos

Los atributos `ISR( )` proporcionan más instrucciones al compilador sobre cómo configurar la función de interrupción.

## ISR\_BLOCK

Las interrupciones globales son inicialmente deshabilitadas por el hardware AVR al ingresar al ISR. Esta configuración **no modifica** este estado.



Este atributo es **idéntico a una macro ISR( ) sin atributo especificado**

## ISR\_NOBLOCK

ISR se ejecuta con interrupciones globales habilitadas inicialmente. El compilador activa el indicador de habilitación de interrupción lo antes posible dentro de la ISR para garantizar un retraso de procesamiento mínimo para las interrupciones anidadas.



Esto se puede usar para crear ISR anidados, sin embargo, se debe tener cuidado para evitar desbordamientos de pila o para evitar ingresar infinitamente al ISR en aquellos casos en los que el hardware AVR no borre el indicador de interrupción respectivo antes de ingresar al ISR.

## ISR\_NAKED

ISR se crea sin código de prólogo o epílogo. El código de usuario es responsable de la preservación del estado de la máquina, incluido el registro SREG, así como de colocar un `reti()` al final de la rutina de interrupción.

## ISR\_ALIASOF(id\_vector)

Esto se puede usar para definir vectores adicionales que comparten el mismo controlador. El siguiente ejemplo crea un alias del vector PCINT1 para el controlador PCINT0:



```
1 ISR(PCINT0_vect) ? ▲
2 {                         ▾
3   ...
4   // Code to handle the event
5 }
6 ISR(PCINT1_vect, ISR_ALIASOF( ▾

```

## Ejemplo de ISR( )

En este ejemplo de código, destacamos los archivos de encabezado requeridos y la definición ISR correcta de una función de controlador para la fuente de interrupción del modo Timer/Counter1 Clear-Timer-On-Compare (CTC). El controlador alterna **LED0** en el **ATmega328PB Xplained Mini (/boards:atavr328)** cada 100 mS:



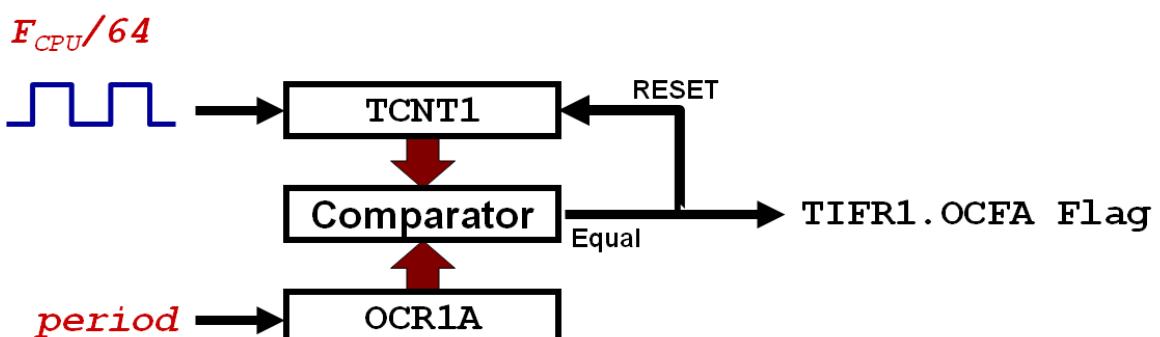
```

1 #include <avr/io.h>?
2 #include <avr/interru
3
4 ISR(TIMER1_COMPA_vect)
5 {
6     PORTB ^= (1 << PC
7 }
8
9 int main(void)
10 {
11     // Initialization
12
13     // Set LED as out
14     DDRB |= (1 << POF
15     PORTB &= ~(1 << P
diecisés
17     // Set up Timer/0
18     TCCR1B |= (1 << W
19     OCR1A = 25000;
20
21     TIMSK1 |= (1 << C
22     TCCR1B |= ((1 <<
23
24     // Enable all int
25     sei();
26
27     while(1);
28 }
```

## Paso 3. Configurar el periférico

A continuación, debe configurar el periférico para generar eventos de solicitud de interrupción.

Por ejemplo, el ATmega328PB contiene varios módulos periféricos de temporizador/contador. Cada módulo tiene un modo llamado **Clear Timer on Compare** (CTC) que, cuando se inicializa correctamente, activará periódicamente una señal de **indicador de coincidencia de comparación de salida del temporizador 1** en el registro de indicador de interrupción TIFR1, como se muestra:



(/local--files/8avr:interrupts-mega-configuration/timer1-ctc-int-flag.png)

En este ejemplo, inicializaremos Timer/Counter1 en modo CTC para generar solicitudes de interrupción cada 100 mS, dada una entrada preescala de 250 kHz (16 MHz/64):



```

1 #include <avr/io.h>?
2 #include <avr/interruption.h>
3
4 ISR(TIMER1_COMPA_vect)
5 {
6     PORTB ^= (1 << PC0);
7 }
8
9 int main(void)
10 {
11     // Initialization
12
13     // Set LED as output
14     DDRB |= (1 << POF0);
15     PORTB &= ~(1 << POF0);
16
17     // Set up Timer/Counter 1
18     TCCR1B |= (1 << WGM12);
19     OCR1A = 25000;
20
21     TIMSK1 |= (1 << OCIE1A);
22     TCCR1B |= ((1 << CS10) | CS11);
23
24     // Enable all interrupts
25     sei();
26
27     while(1);
28 }
```



Este es un ejemplo de una interrupción no persistente (/avr:interrupts-mega-overview#non-persistent-interrupts) . El indicador TIFR1.OCFA se borra automáticamente por el hardware al ingresar al controlador.



El indicador TIFR1.OCFA también se puede borrar manualmente escribiendo un "1" lógico en la ubicación del bit.

## Paso 4. Habilitar todas las interrupciones

Finalmente, debemos habilitar globalmente todas las interrupciones periféricas habilitadas configurando el **bit I de habilitación de interrupción global** en el **registro de estado (SREG)** . La biblioteca de interrupciones AVR-LIBC proporciona dos funciones de macro útiles para esto:

- `sei()` para habilitar interrupciones globalmente
- `cli()` para deshabilitar las interrupciones globalmente



```

1  #include <avr/io.h>?
2  #include <avr/interruptions.h>
3
4  ISR(TIMER1_COMPA_vect)
5  {
6      PORTB ^= (1 << PC0);
7  }
8
9  int main(void)
10 {
11     // Initialization
12
13     // Set LED as output
14     DDRB |= (1 << POF0);
15     PORTB &= ~(1 << POF0);
16
17     // Set up Timer/Counter 1
18     TCCR1B |= (1 << CS10);
19     OCR1A = 25000;
20
21     TIMSK1 |= (1 << OCIE1A);
22     TCCR1B |= ((1 << OCIE1A) & 0x01);
23
24     // Enable all interrupts
25     sei();
26
27     while(1);
28 }
```

## Aprende más



### Resumen de interrupciones de megaAVR

Más información > ([/8avr:interrupts-mega-overview](#))



### Consideraciones especiales

Más información > ([/8avr:interrupts-special-considerations](#))



### Ejemplo de interrupción de megaAVR

Más información > ([/8avr:interrupts-mega-example](#))

# Consideraciones Especiales

Esta página cubre algunas consideraciones especiales a tener en cuenta cuando se trabaja con interrupciones en MCU AVR.

## Compartir datos con el ISR

Las variables compartidas entre el ISR y el programa principal deben declararse como **volátiles** y tener un alcance **global**. Al compilar usando el optimizador, en un bucle como el siguiente:



```
1  uint8_t flag; ?  
2  ...  
3  ISR(SOME_vect) {  
4      flag = 1;  
5  }  
6  ...  
7  while(flag == 0) {  
8  ...  
9  }
```

el compilador generalmente accederá a "bandera" solo una vez y optimizará los accesos adicionales por completo, ya que su análisis de ruta de código muestra que nada dentro del ciclo podría cambiar el valor de "bandera" de todos modos. Para decirle al compilador que esta variable podría cambiarse fuera del alcance de su análisis de ruta de código (por ejemplo, dentro de una rutina de servicio de interrupción), la variable debe declararse así:



```
1  volatile uint8_t flag; ?  
2  ...  
3  ISR(SOME_vect) {  
4      flag = 1;  
5  }  
6  ...  
7  while(flag == 0) {  
8  ...  
9  }
```

Cuando la variable se declara **volátil** como se indicó anteriormente, el compilador se asegura de que dondequiera que se actualice o lea la variable, siempre escribirá los cambios en la memoria SRAM y leerá la variable desde SRAM.

## Operaciones de datos atómicos

Para que una operación sea considerada **atómica**, debe garantizar el acceso **ininterrumpido** de una determinada variable. Muchos lenguajes ensambladores brindan esto en ciertos niveles, es decir, prueba y configuración de bits, sin embargo, no existe **ninguna disposición** para proporcionar automáticamente la atomicidad de todos los tipos de variables en el lenguaje ANSI C.



Las expresiones y declaraciones ANSI-C **no son atómicas**.

Este problema puede ser problemático (en ciertas situaciones) cuando **las variables de varios bytes se comparten** con un ISR. Si bien declarar una variable de este tipo como **volátil** garantiza que el compilador no optimizará los accesos a ella, no garantiza el acceso **atómico** a ella. Considere el siguiente ejemplo de código:



```
1 #include <stdint.h> ?
2 #include <avr/io.h>
3 #include <avr/interrupt>
4
5 volatile uint16_t ctr;
6
7 ISR(TIMER1_OVF_vect)
8 {
9     ctr--;
10 }
11 ...
12 int
13 main(void)
14 {
15     ...
16     ctr = 0x0200;
17     start_timer();
18     while(ctr != 0)
19         // wait
20         ;
21     ...
22 }
```



Existe la posibilidad de que el contexto principal salga de su ciclo `while()` cuando la variable `ctr` alcance el valor 0x00FF. Esto sucede porque el compilador no puede acceder de forma nativa a una variable de 16 bits de forma atómica en una CPU de 8

bits. Entonces, cuando **ctr** está, por ejemplo, en 0x0100, el compilador luego prueba el byte bajo para 0, lo que tiene éxito. Luego procede a probar el byte alto, pero en ese momento se activa el ISR y el contexto principal se interrumpe. El ISR disminuirá la variable de 0x0100 a 0x00FF, luego continúa el contexto principal. Ahora prueba el byte alto de la variable que (ahora) también es 0, por lo que concluye que la variable ha llegado a 0 y finaliza el ciclo.

## Macros de acceso atómico

La biblioteca atómica ([http://www.nongnu.org/avr-libc/user-manual/group\\_\\_util\\_\\_atomic.html](http://www.nongnu.org/avr-libc/user-manual/group__util__atomic.html)) AVR-LIBC proporciona las macros **ATOMIC\_BLOCK** que insertan la protección de interrupción adecuada cuando se desea acceso atómico. Estas macros operan a través de la manipulación automática del bit de **estado de interrupción global (I) del registro SREG**. Las rutas de salida de ambos tipos de bloques se gestionan automáticamente sin necesidad de consideraciones especiales, es decir, el estado de interrupción se restaurará al mismo valor que tenía al entrar en el bloque respectivo. Usando las macros de este archivo de encabezado, el código anterior se puede reescribir como:



```
1 #include <stdint.h>?
2 #include <avr/io.h>
3 #include <avr/interrupt.h>
4 #include <util/atomic.h>
5
6 volatile uint16_t ctr;
7
8 ISR(TIMER1_OVF_vect)
9 {
10     ctr--;
11 }
12 ...
13 int main(void)
14 {
15     uint16_t ctr_copy;
16     ...
17     ctr = 0x0200;
18     start_timer();
19     do
20     {
21         ATOMIC_BLOCK(ATOMIC_BLOCK)
22         {
23             ctr_copy = ctr;
24         }
25     } while(ctr != 0);
26     // wait
27     ;
28 }
29 ...
```

La macro **ATOMIC\_BLOCK** instalará la protección de interrupción adecuada antes de acceder a la variable **ctr** , por lo que se garantiza que se probará de manera consistente. En este caso, el parámetro **ATOMIC\_RESTORESTATE** hace que **ATOMIC\_BLOCK** restaure el estado anterior del registro SREG, guardado antes de que se deshabilitara el bit indicador de estado de interrupción global. El efecto neto de esto es hacer que el contenido de **ATOMIC\_BLOCK** sea atómico garantizado, sin cambiar el estado del indicador de estado de interrupción global cuando se completa la ejecución del bloque.

## Aprende más



### Resumen de interrupciones de megaAVR®

Más información > ([/8avr:interrupts-mega-overview](#))



### Configuración de interrupciones megaAVR®

Más información > ([/8avr:interrupts-mega-configuration](#))



### Ejemplo de interrupción de megaAVR®

Más información > ([/8avr:interrupts-mega-example](#))

# Ejemplo de código de interrupción megaAVR®

## ⌚ Objetivo

Esta página proporciona un ejemplo de código de **interrupción** básico para la MCU **ATmega328PB** . (<http://www.microchip.com/wwwproducts/en/ATmega328PB>) El proyecto configura el módulo Timer/Counter1 para operar en modo **Clear-Timer-On-Compare** (CTC) y, en una coincidencia de período, genera un evento de interrupción cada 100 mS. El ISR manipula una variable de señal de "marca" que utiliza el bucle principal para alternar LED0 cada 100 mS.

## ☑ Materiales

### Herramientas de ferretería

Herramienta	Sobre
 ( <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">http://www.atmel.com/tools/MEGA328PB-XMINI.aspx</a> )	Mini kit de evaluación ATmega328PB Xplained <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">(http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)</a> <a href="https://www.microchipdirect.com/Products/Product.aspx?ProdID=1740000">(https://www.microchipdirect.com/Products/Product.aspx?ProdID=1740000)</a>

### Herramientas de software

Herramienta	Sobre	Instaladores			Instrucciones de instalación
		 ventanas	 linux	 Mac OS X	
 Entorno de desarrollo integrado Atmel® Studio	<a href="http://atstudio:start">(/atstudio:start)</a> <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe</a>	 <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe</a>	 <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe</a>	 <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe</a>	<a href="http://install:atstudio">(/install:atstudio)</a>

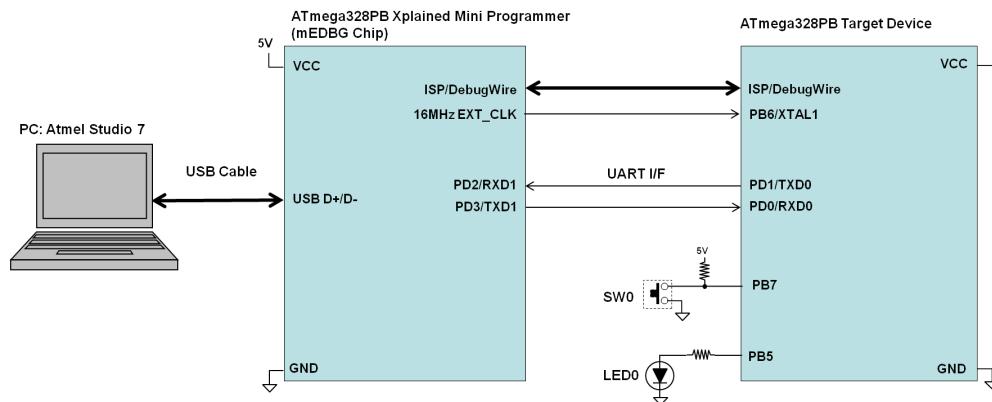
### Archivos de ejercicios

	Descargar	
Expediente	 ventanas	 linux
 Proyecto de ejemplo	<a href="http://local--files/8avr:interrupts-mega-example/8avr-mega-int-usage.zip">http://local--files/8avr:interrupts-mega-example/8avr-mega-int-usage.zip</a>	<a href="http://local--files/8avr:interrupts-mega-example/8avr-mega-int-usage.zip">http://local--files/8avr:interrupts-mega-example/8avr-mega-int-usage.zip</a>

 Recomendamos extraer el archivo .zip a su carpeta C:\. Debería ver la carpeta C:\MTT\8avr\mega\code-examples\interrupt-example\8avr-mega-int-usage que contiene la solución 8avr-mega-int-usage.atsln

## ⚡ Diagrama de conexión

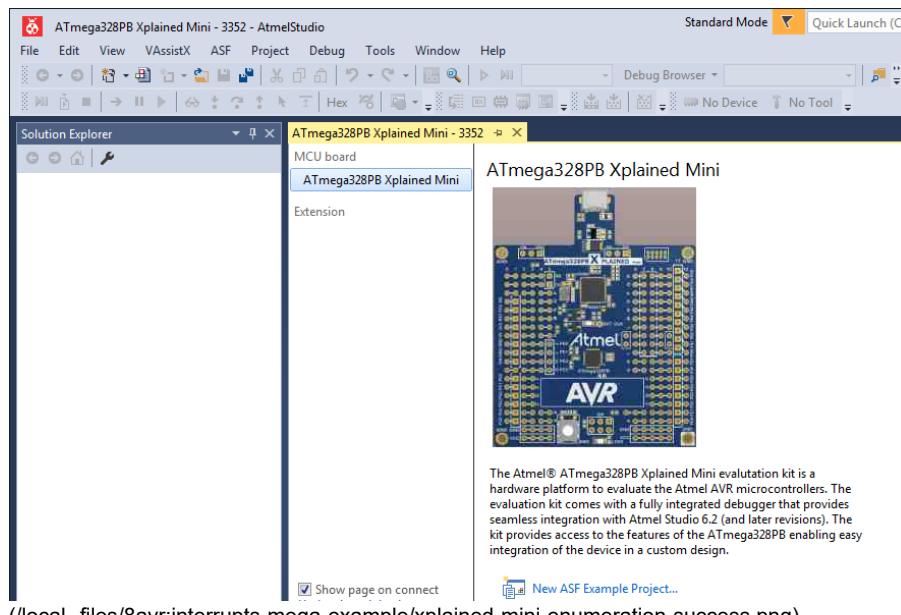
El módulo USART0 en el dispositivo ATmega328PB de destino está conectado a la interfaz USART en el chip mEDBG. El chip mEDBG realiza la conversión USB-serie enumerando como un puerto COM virtual de clase CDC en la PC y presentando los datos USART de destino en esta interfaz. El mEDBG también controla la interfaz de programación/depuración en el dispositivo de destino, además de proporcionar un reloj de 16 MHz cuando la placa Xplained está conectada mediante un cable USB a una PC. El LED0 está conectado al puerto PB5 como se muestra:



(/local--files/8avr:interrupts-mega-example/xplained-mini-connection-diagram-as7.png)

## Procedimiento

Conecte la mini placa Xplained ATmega328PB a su computadora usando un cable USB A-a-MicroB. Inicie Atmel Studio 7. Si la placa se ha enumerado correctamente, debería ver la imagen de la placa en Studio como se muestra:

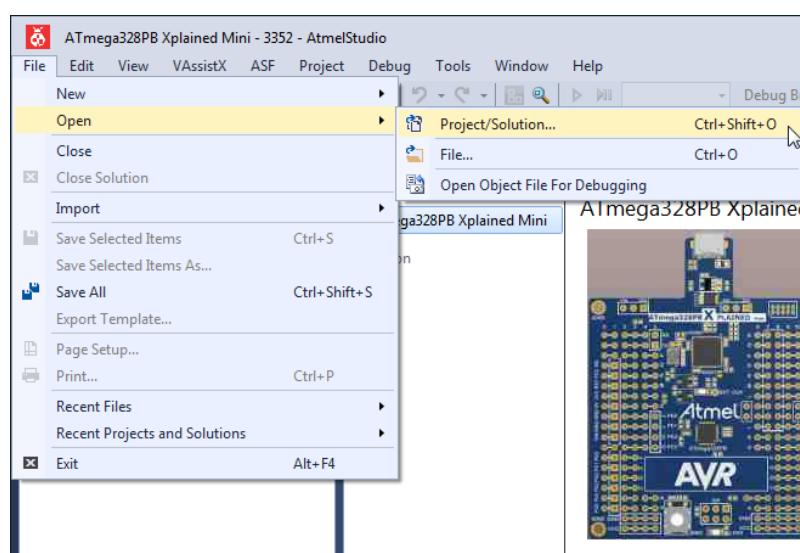


(/local--files/8avr:interrupts-mega-example/xplained-mini-enumeration-success.png)

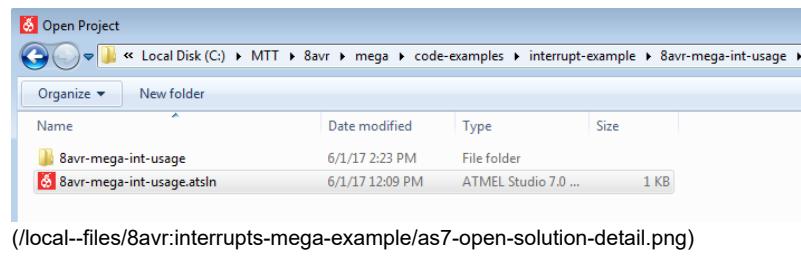


La placa se identifica por los últimos cuatro dígitos de su número de serie (consulte la etiqueta en la parte inferior de la placa). En el ejemplo anterior, los últimos cuatro dígitos son "3352"

### 1 Abra la solución

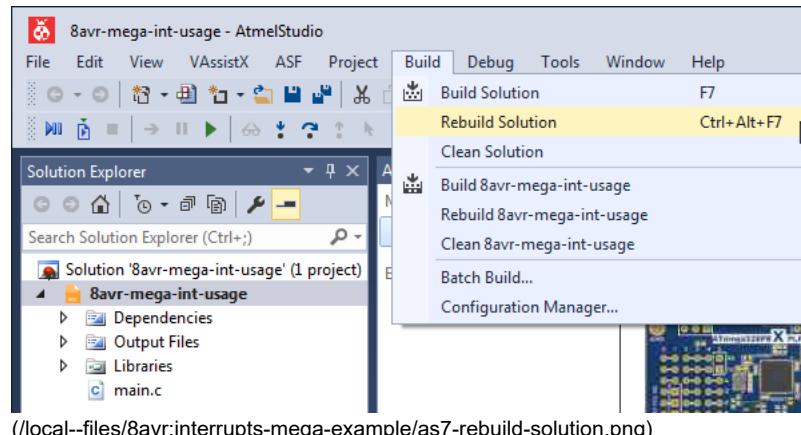


(/local--files/8avr:interrupts-mega-example/as7-open-solution.png)



Para comprender cómo se configuraron y habilitaron las interrupciones en este ejemplo ( archivo `main.c` ), consulte la página **Configuración de interrupciones del megaAVR®** ([/8avr:interrupts-mega-configuration](#)) .

## 2 Reconstruir la solución

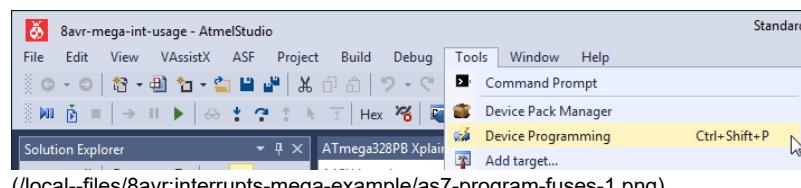


## 3 Programa los fusibles

Hay varios ajustes de configuración de hardware clave que deben configurarse. Los siguientes **ajustes** ([/8avr:avrfuses](#)) de fusibles deben programarse en el dispositivo:

- ALTO: 0xDF
- BAJO: 0xC0
- EXT: 0xFC

Ingrese al cuadro de diálogo Programación del dispositivo como se muestra:



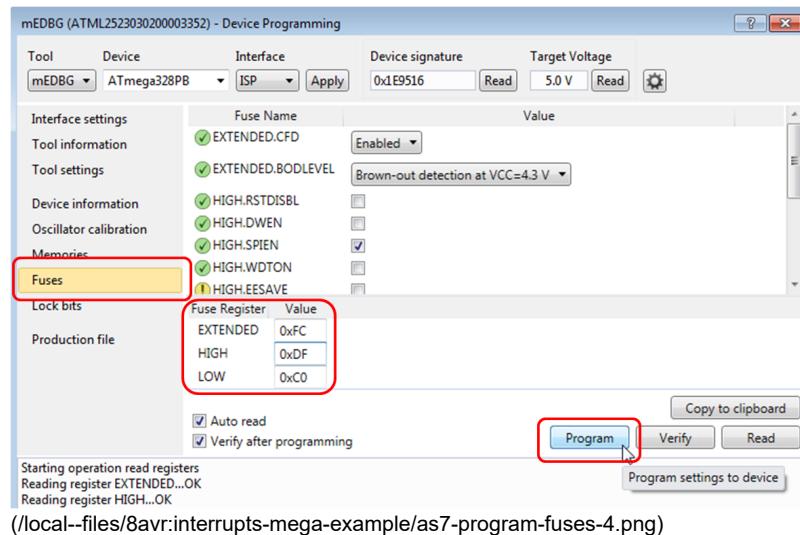
En el cuadro de diálogo Programación de dispositivos, seleccione la **herramienta** , el **dispositivo** y la **interfaz** como se muestra, luego presione **Aplicar** :



Para verificar una conexión, seleccione **Leer** y verifique que se encuentre una **firma de dispositivo** :

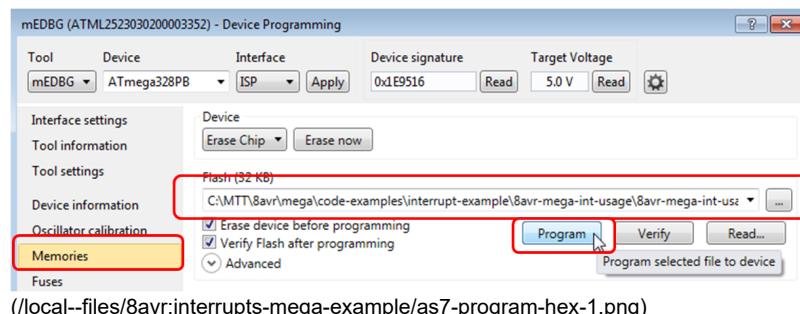


Seleccione la subsección **Fusibles** , ingrese los 3 valores de byte del fusible arriba, luego presione **Programar** como se muestra:

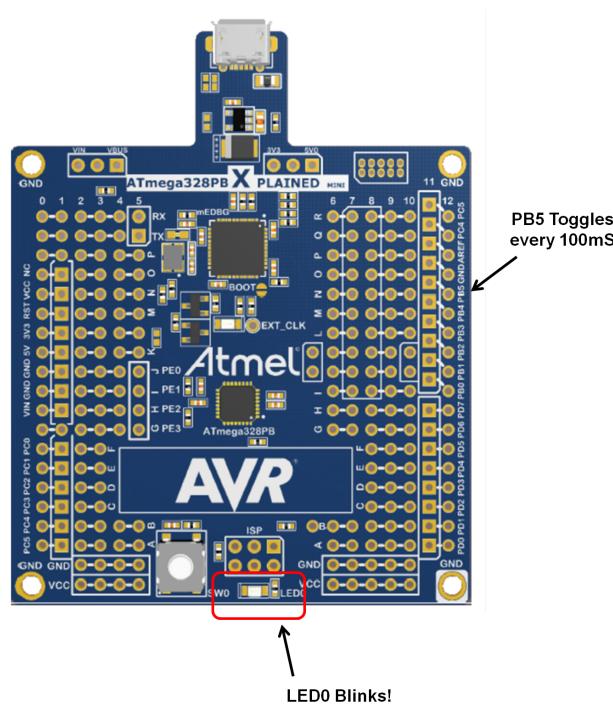


#### 4 Programa el archivo hexadecimal

Mientras aún está en el cuadro de diálogo Programación del dispositivo, seleccione "Memorias" como se muestra. La ruta al archivo hexadecimal de la solución ya debería aparecer en el cuadro de diálogo. Presione Programa como se muestra:



## ★ Resultados



## 💡 Conclusiones

Este proyecto ha proporcionado un ejemplo de cómo configurar y usar interrupciones en la MCU megaAVR.

## Aprende más



### **Resumen de interrupciones de megaAVR**

Más información > ([/8avr:interrupts-mega-overview](#))



### **Configuración de interrupciones megaAVR**

Más información > ([/8avr:interrupts-mega-configuration](#))



### **Consideraciones especiales**

Más información > ([/8avr:interrupts-special-considerations](#))

# Resumen de interrupciones de megaAVR®

La familia megaAVR® proporciona varias fuentes de interrupción diferentes, todas las cuales son enmascarables y se dividen en tres categorías:

- **Interrupciones periféricas internas**
  - Asociado con temporizadores, USART, SPI, periféricos ADC
- **Interrupciones de clavijas externas**
  - Asociado con los pines de interrupción externa INT0-INT7
- **Interrupciones de cambio de pin**
  - Asociado con interrupciones externas PCINT0-PCINT2 que ocurren en un cambio de pin de puerto

A los periféricos se les asignan **bits de habilitación de interrupción** individuales en su respectivo **registro de máscara de interrupción** que debe escribirse como uno lógico junto con el **bit I de habilitación de interrupción global** en el **registro** de estado para habilitar la interrupción.

**Name:** SREG  
**Offset:** 0x5F  
**Reset:** 0x00  
**Property:** When addressing as I/O Register: address offset is 0x3F

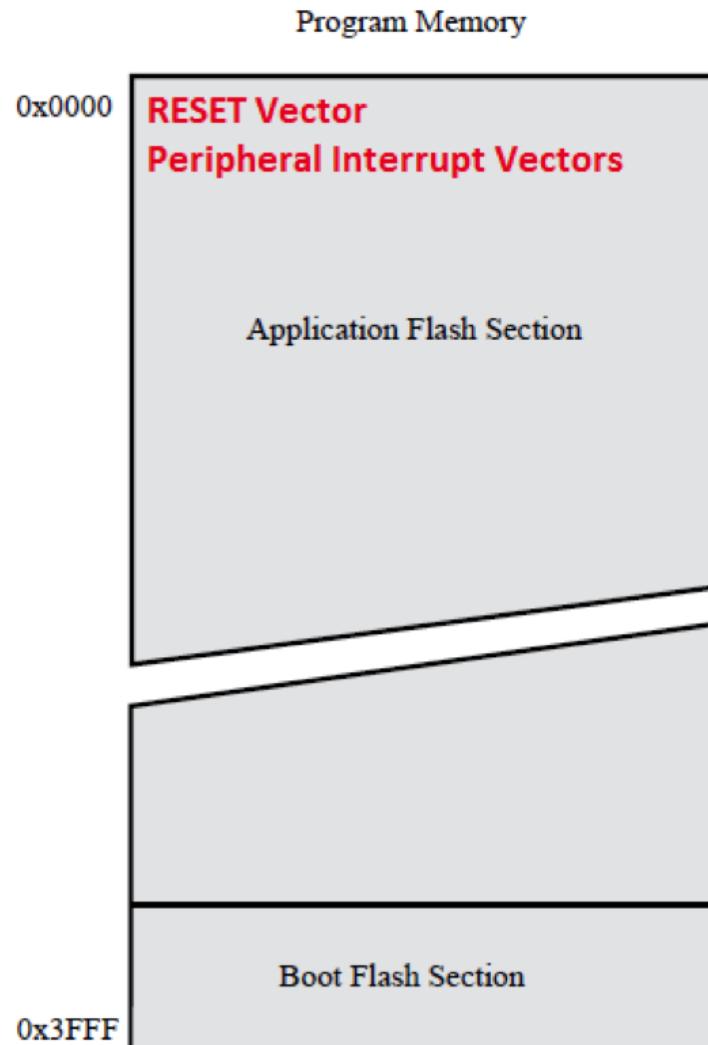
Bit	7	6	5	4	3	2	1	0
Access	I	T	H	S	V	N	Z	C
Reset	R/W							

(/local--files/8avr:interrupts-mega-overview/status-register.png)

## Restablecer e interrumpir ubicaciones de vectores

Cada una de las fuentes de reinicio e interrupción tiene un vector de programa separado en el **espacio de memoria** del programa. Las direcciones más bajas en el espacio de la memoria del programa se definen de manera predeterminada como

vectores de reinicio e interrupción, como se muestra:



(/local--files/8avr:interrupts-mega-overview/vectors.png)

## Reubicación de vectores

El usuario puede reubicar el vector RESET así como la ubicación de inicio de los vectores de Interrupción en la **Sección Flash de Arranque** del espacio de la memoria del programa programando el bit de fusible **BOOTRST** en "0" y configurando el bit **IVSEL** del Registro de Configuración del Microcontrolador ( **MCUCR** ) en "1". Aquí se muestra la posible ubicación del vector de interrupción y RESET:

<b>BOOTRST</b>	<b>IVSEL</b>	<b>Restablecer dir.</b>	<b>Dirección de inicio del vector de interrupción.</b>
1	0	0x0000	0x0002
1	1	0x0000	Dirección de reinicio de arranque + 0x0002

0	0	Dirección de reinicio de arranque	0x0002
0	1	Dirección de reinicio de arranque	Dirección de reinicio de arranque + 0x0002

La **dirección de reinicio de arranque** se establece mediante bits de fusible BOOTSZ0/BOOTSZ1 como se muestra aquí para ATmega328PB:

Table 32-7 Boot Size Configuration, ATmega328PB

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	256 words	4	0x0000 - 0x3EFF	0x3F00 - 0x3FFF	0x3EFF	0x3F00
1	0	512 words	8	0x0000 - 0x3DFF	0x3E00 - 0x3FFF	0x3DFF	0x3E00
0	1	1024 words	16	0x0000 - 0x3BFF	0x3C00 - 0x3FFF	0x3BFF	0x3C00
0	0	2048 words	32	0x0000 - 0x37FF	0x3800 - 0x3FFF	0x37FF	0x3800

(/local--files/8avr:interrupts-mega-overview/bootflash328pb.png)



Los fusibles se programan utilizando un **procedimiento de programación especial** (**/8avr:avrFuses**) dentro de Atmel Studio 7 u otro programador.



Para evitar cambios no intencionales de las tablas de vectores de interrupción, se debe seguir un procedimiento de escritura especial para cambiar el bit IVSEL:

- Escriba el bit de habilitación de cambio de vector de interrupción (IVCE) a uno.
- Dentro de cuatro ciclos, escriba el valor deseado en IVSEL mientras escribe un cero en IVCE.

Aquí hay un ejemplo de código que muestra cómo modificar el bit IVSEL y reubicar los vectores de interrupción:



```

1  <font></font> ?
2  void move_interrupts(void)<font>?
3  {<font></font>
4      uchar temp;<font></font>
5      /* GET MCUCR */<font></font><font>
6      temp = MCUCR;<font></font>
7      /* Enable change of Interrup
8      MCUCR = temp | (1 << IVCE);<
9      /* Move interrupts to Boot !
10     MCUCR = temp | (1 << IVSEL);<
11 }<font></font>
12 <font></font>

```

# Nivel de prioridad

Cada vector tiene un nivel de prioridad predeterminado: cuanto **menor** sea la dirección, **mayor** será el nivel de prioridad. RESET tiene la prioridad más alta, y el siguiente es INT0: la solicitud de interrupción externa 0. El siguiente gráfico muestra la lista de vectores parciales para la MCU ATmega328PB:

Table 16-1 Reset and Interrupt Vectors in ATmega328PB

Vector No	Program Address	Source	Interrupts definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
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	SPI0_STC	SPI1 Serial Transfer Complete
19	0x0024	USART0_RX	USART0 Rx Complete
20	0x0026	USART0_UDRE	USART0, Data Register Empty
21	0x0028	USART0_TX	USART0, Tx Complete
22	0x002A	ADC	ADC Conversion Complete

(/local--files/8avr:interrupts-mega-overview/vectors328pb.png)

# Procesamiento de interrupciones

Cuando ocurre una interrupción, el bit I de habilitación de interrupción global se borra y todas las interrupciones se desactivan. El bit I se establece automáticamente cuando se ejecuta una instrucción Return from Interrupt (RETI).



El software del usuario puede escribir uno lógico en el bit I para habilitar **interrupciones anidadas**. Todas las interrupciones habilitadas pueden interrumpir la rutina de interrupción actual.

Hay básicamente dos tipos de interrupciones:

## Interrupciones persistentes

Este tipo de interrupción se activará siempre que la condición de interrupción esté presente. Estas interrupciones no necesariamente tienen banderas de interrupción.

**Ejemplo: Interrupción de recepción completa de USART** El USART contiene un indicador de recepción completa (RXC) que se establece si hay datos no leídos en el búfer de recepción. Cuando se establece la habilitación de interrupción de recepción completa (RXCIE) en UCSRnB, la interrupción de recepción completa de USART se ejecutará siempre que el indicador RXC esté establecido (siempre que las interrupciones globales estén habilitadas). Cuando se utiliza la recepción de datos impulsada por interrupciones, la rutina de recepción completa debe leer los datos recibidos de UDR para borrar el indicador RXC; de lo contrario, se producirá una nueva interrupción una vez que finalice la rutina de interrupción.

## Non-Persistent Interrupts

Este tipo de interrupción se desencadena por un evento que establece un **indicador de interrupción** . Para estas interrupciones, el contador de programa se vectoriza al vector de interrupción real para ejecutar la rutina de manejo de interrupciones, y **el hardware borra el indicador de interrupción correspondiente** .. Las banderas de interrupción también se pueden borrar escribiendo un uno lógico en la(s) posición(es) del bit de bandera que se va a borrar. Si se produce una condición de interrupción mientras se borra el bit de activación de interrupción correspondiente, el indicador de interrupción se establecerá y se recordará hasta que se habilite la interrupción o el software borre el indicador. De manera similar, si ocurren una o más condiciones de interrupción mientras se borra el bit de habilitación de interrupción global, los indicadores de interrupción correspondientes se establecerán y recordarán hasta que se establezca el bit de habilitación de interrupción global, y luego se ejecutarán por orden de prioridad. **Ejemplo: Timer/Counter0 Overflow Interrupt** Bit-0 del Timer0 Interrupt Flag Register (TIFR0) contiene el indicador de interrupción TOV0. Este indicador se establece cuando se produce un desbordamiento en Timer/Counter0. TOV0 es

**borrado por el hardware al ejecutar el vector de manejo de interrupción correspondiente** . Alternativamente, TOV0 se borra escribiendo un uno lógico en la bandera. Cuando se establecen el bit I de SREG, TOIE0 (habilitación de interrupción de desbordamiento del temporizador/contador0) y TOV0, se ejecuta la interrupción de desbordamiento del temporizador/contador0.

## Aprende más



### **Configuración de interrupciones megaAVR**

Más información > (/8avr:interrupts-mega-configuration)



### **Consideraciones especiales**

Más información > (/8avr:interrupts-special-considerations)



### **Ejemplo de interrupción de megaAVR**

Más información > (/8avr:interrupts-mega-example)

# Comparador analógico y ejemplo de referencia de voltaje

Este proyecto práctico demuestra un ejemplo simple de comparador analógico y voltaje de referencia: referencia de banda prohibida como entrada positiva (1,1 V) y AIN1 como entrada negativa. Al cablear AIN1 (PD7) al interruptor (SW0, PB7), que se enciende en alto (5 V) y se apaga a GND, y al cablear ACO (PE0) a PB5 (LEDO), la salida del comparador (ACO, PE0) se puede observar comprobando el estado de los LED.

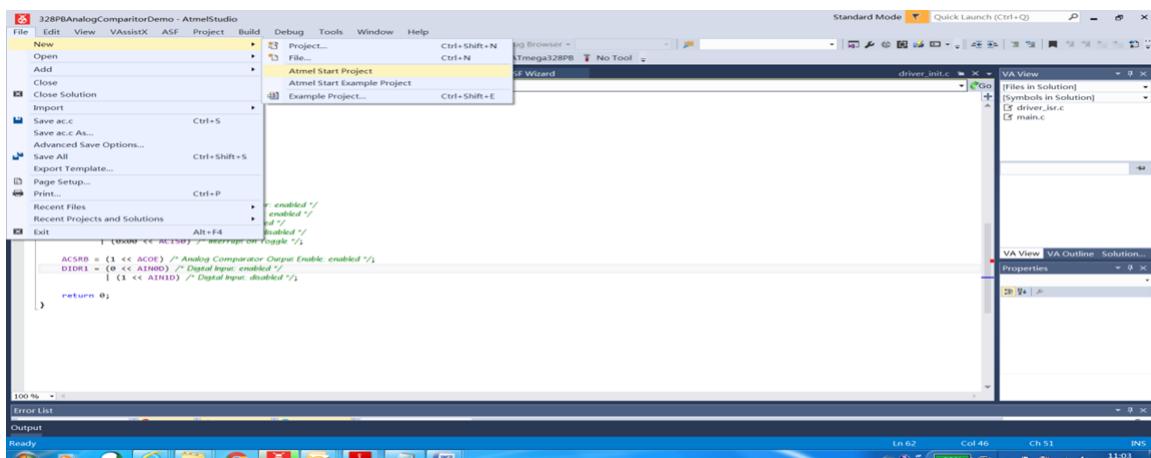
- Cableado : **PD7 (AIN1) | PB7 (SW0; PE0 (ACO) | PB5 (LEDO)**

Tablero explicado:



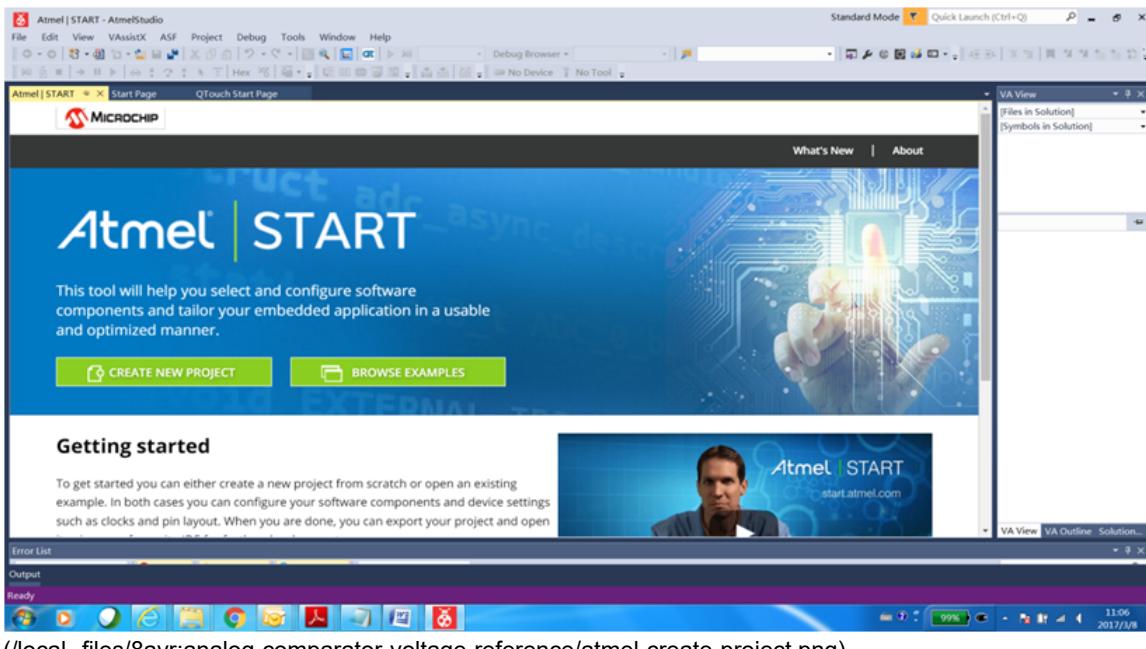
(/local--files/8avr:analog-comparator-voltage-reference/xplained-board.png)

## 1 Nuevo | Proyecto de inicio de Atmel



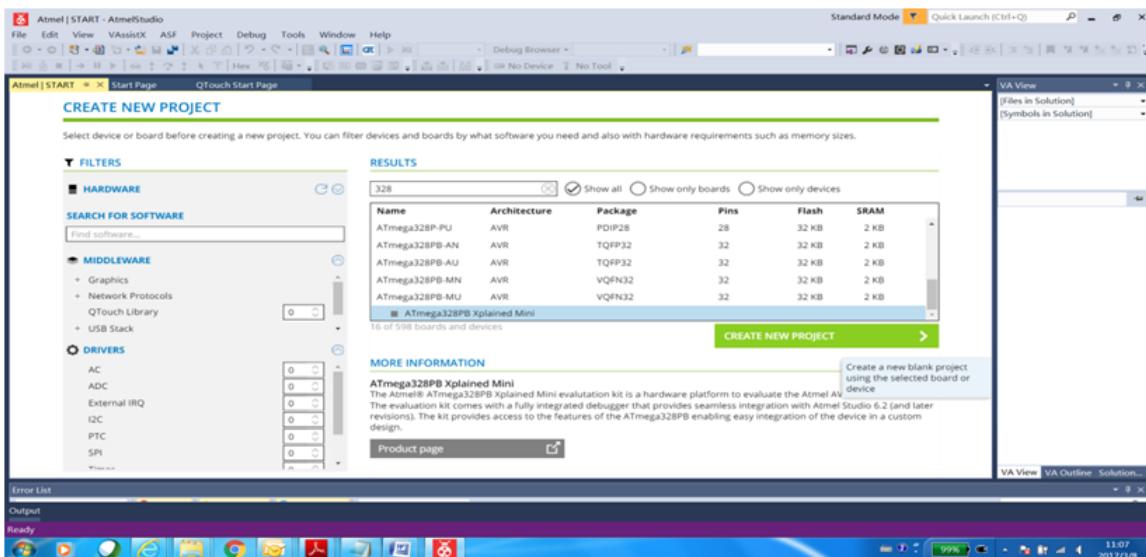
(/local--files/8avr:analog-comparator-voltage-reference/atmel-start-project.png)

## 2 Crear nuevo proyecto



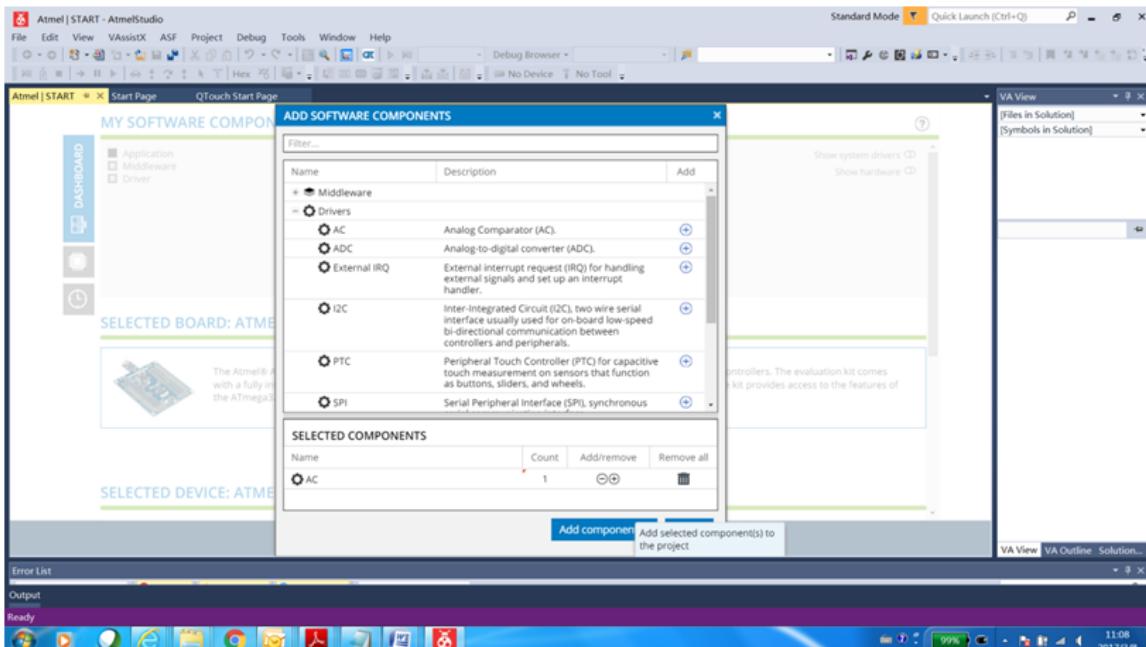
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project.png)

### 3 Seleccione ATmega328PB Mini explicado (/boards:atavr328) :



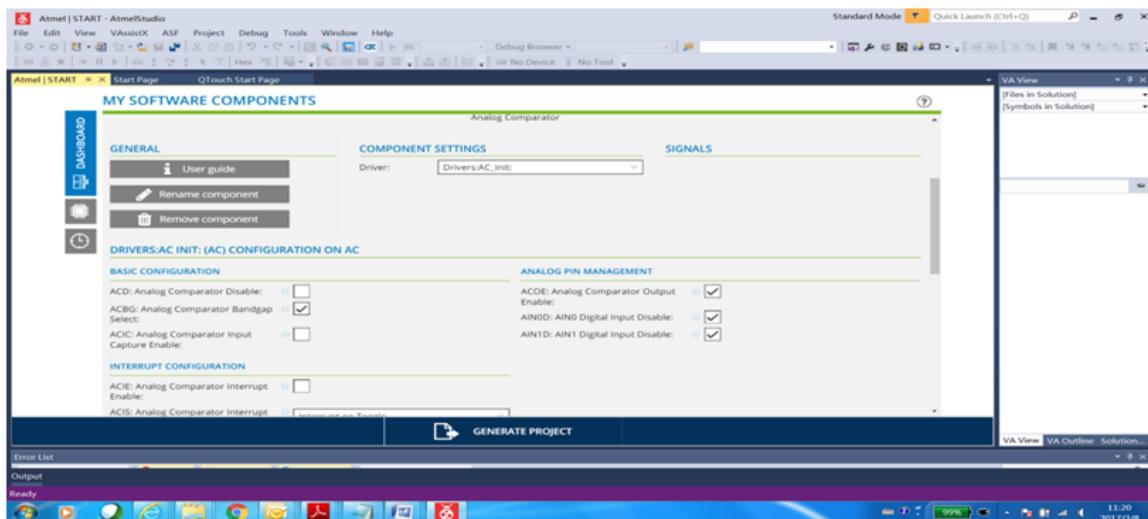
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project-2.png)

### 4 Agregue componentes de software, seleccione AC:



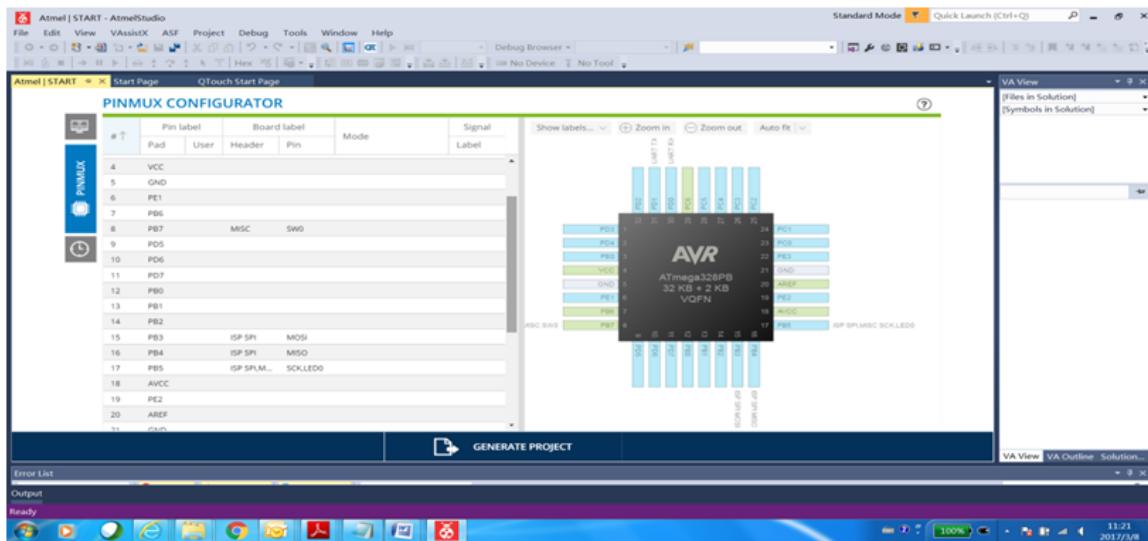
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project-3.png)

## 5 Configuración de CA:



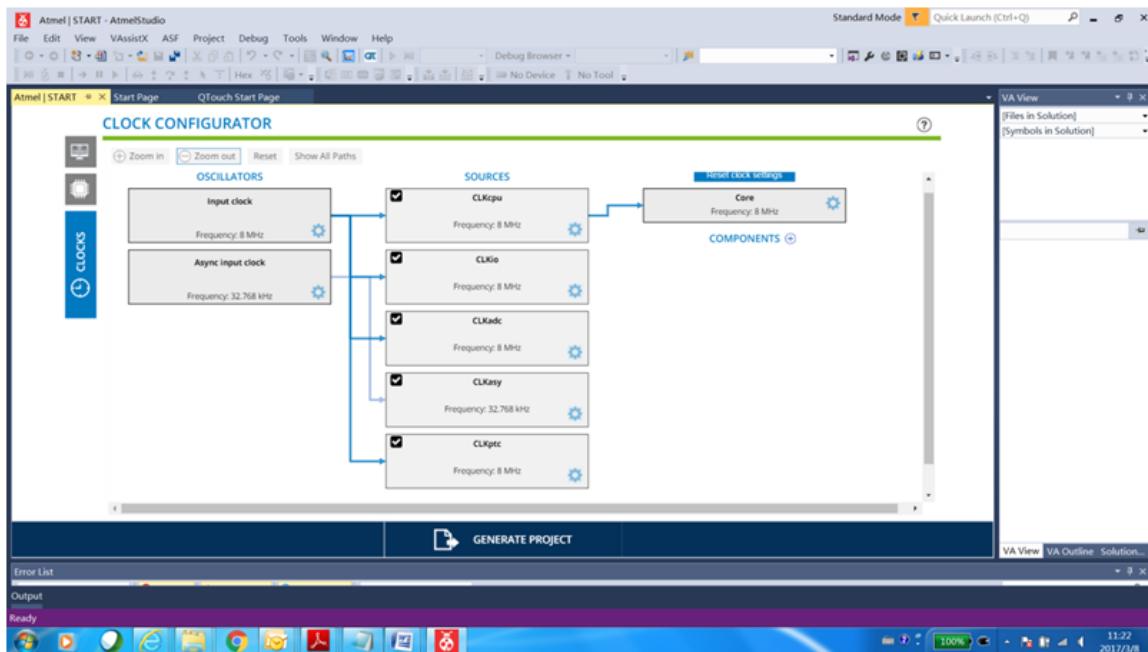
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project-4.png)

## 6 Vista del configurador de pines:



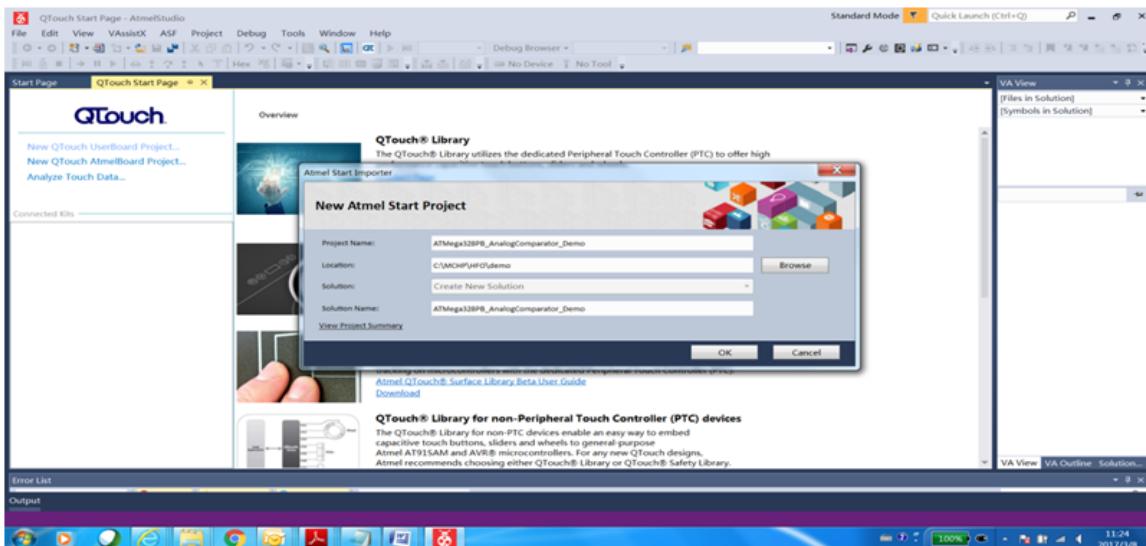
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project-5.png)

## 7 Configuración del configurador de reloj:



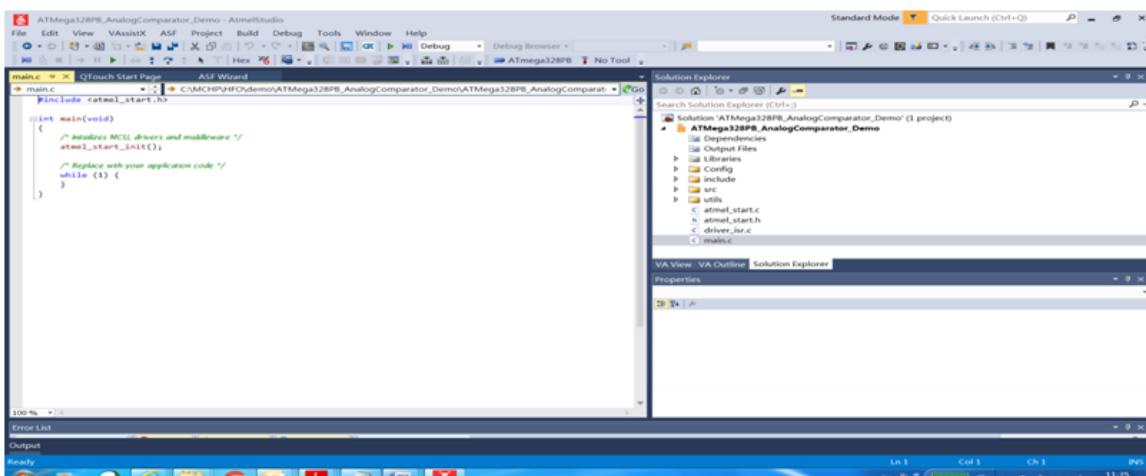
(/local--files/8avr:analog-comparator-voltage-reference/atmel-create-project-6.png)

## 8 Generar Proyecto:



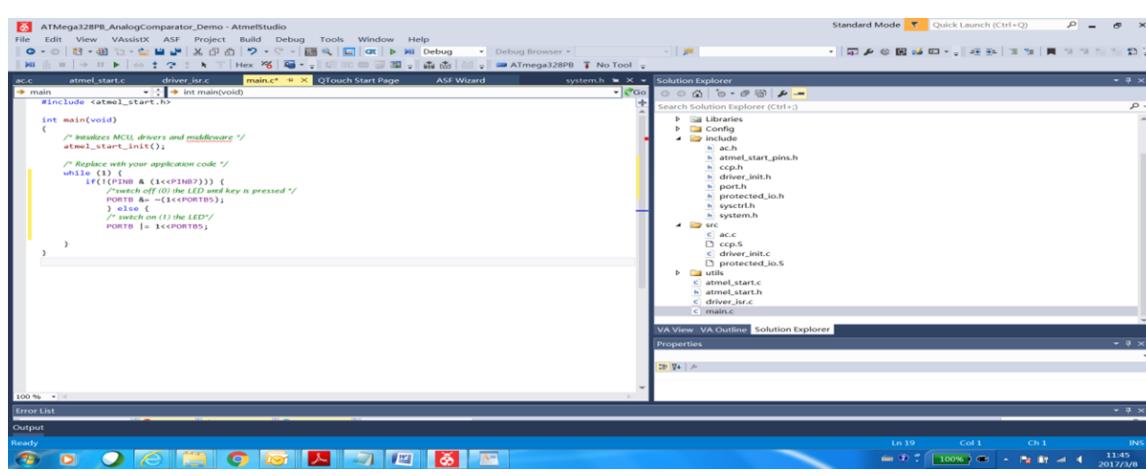
(/local--files/8avr:analog-comparator-voltage-reference/atmel-generate-project.png)

## 9 Ver Proyecto:



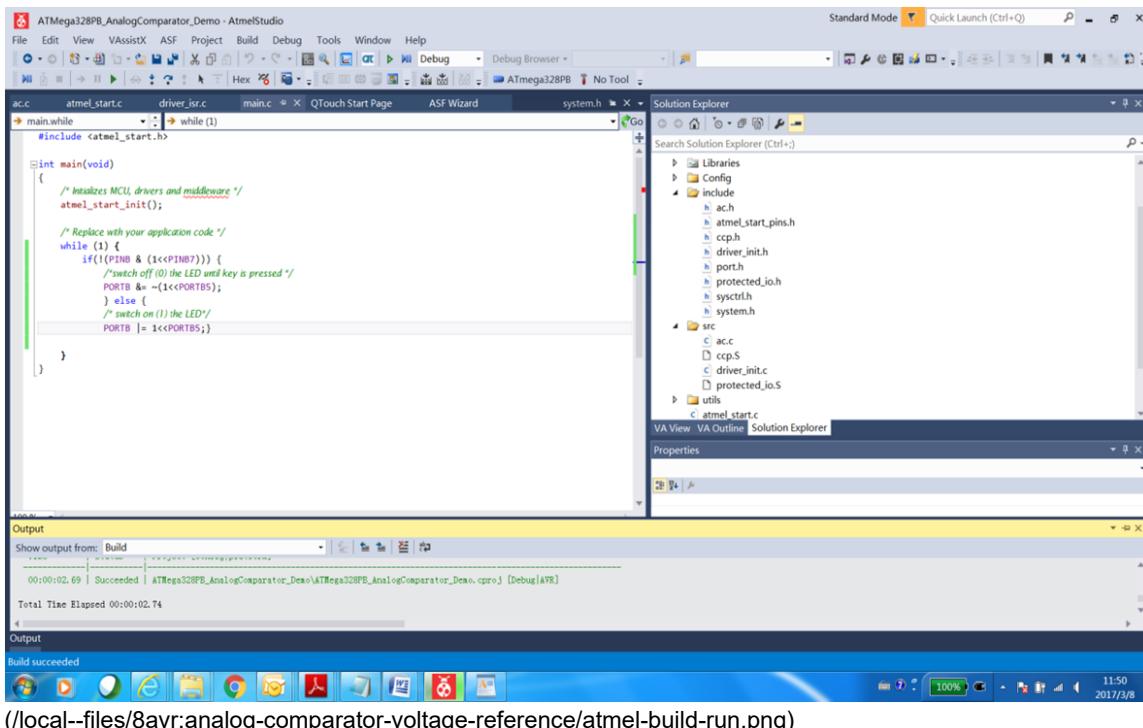
(/local--files/8avr:analog-comparator-voltage-reference/atmel-view-project.png)

## 10 En `Main.c` , agregue escaneo de teclado y controlador LED:



(/local--files/8avr:analog-comparator-voltage-reference/atmel-scan-led-driver.png)

## 11 Construir y ejecutar:



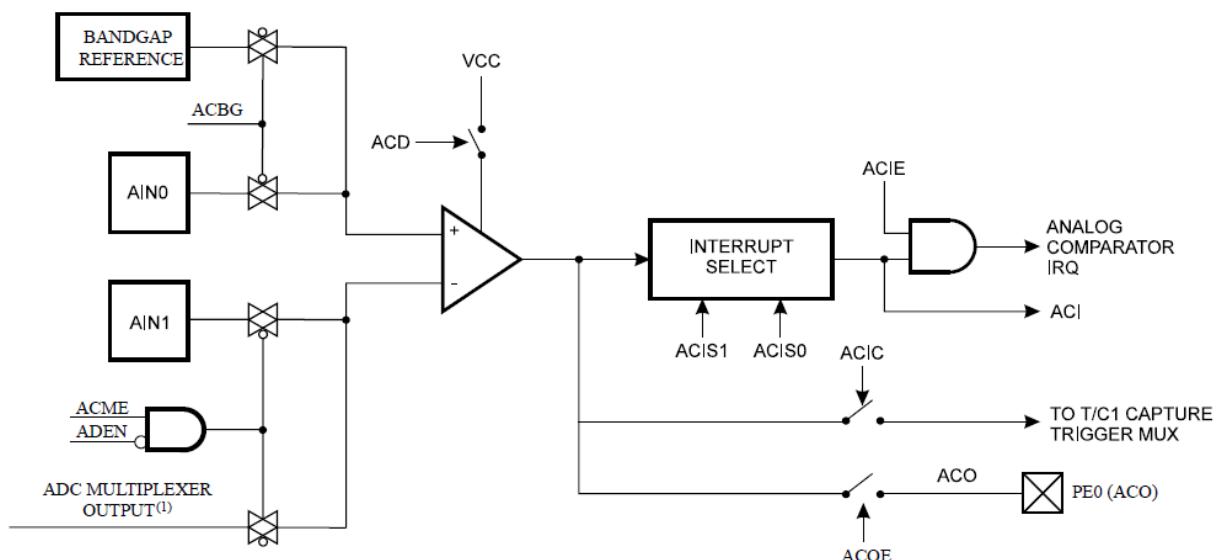
(/local--files/8avr:analog-comparator-voltage-reference/atmel-build-run.png).

Para descargar la demostración del comparador analógico ATmega328PB, visite esta página (<https://microchiptechnology.sharepoint.com/:u/s/DeveloperHelp/EUoeisQUGDNAqXHmQDJm4WgB5lendH3ses4oRBL9zPbrpw?e=AmjbWH>).

# Descripción general de la referencia de voltaje del comparador analógico

El comparador analógico compara los valores de entrada en el pin positivo AIN0 y el pin negativo AIN1. Cuando el voltaje en el pin positivo, AIN0, es mayor que el voltaje en el pin negativo, AIN1, se establece la salida del comparador analógico, ACO (en el puerto E[0]). La salida del comparador se puede configurar para activar la función de captura de entrada del temporizador/contador1. Además, el comparador puede disparar una interrupción separada, exclusiva del comparador analógico. El usuario puede seleccionar la activación de interrupción en el aumento, la caída o la alternancia de la salida del comparador.

Se muestra un diagrama de bloques del comparador y su lógica circundante.



(/local--files/8avr:analog-comparator-voltage-reference/comparator-logic.png)

El bit del convertidor de analógico a digital (ADC) de reducción de potencia en el registro de reducción de potencia (PRR.PRADC) debe escribirse en 0 para poder usar la entrada MIUX de ADC.

## Entrada multiplexada del comparador analógico

Es posible seleccionar cualquiera de los pines ADC[7..0] para reemplazar la entrada negativa al comparador analógico. El multiplexor ADC se utiliza para seleccionar esta entrada y, en consecuencia, el ADC debe apagarse para utilizar esta función. Si el bit de habilitación del multiplexor del comparador analógico en el registro B de control y estado del ADC (ADCSRB.ACME) es uno y el ADC está apagado (ADCSRA.ADEN=0), los tres bits de selección de canal analógico menos significativos en el registro de selección del multiplexor del ADC (ADMUX.MUX[2..0]) seleccione el pin de entrada para reemplazar la entrada negativa al comparador analógico, como se muestra en la siguiente tabla. Cuando ADCSRB.ACME=0 o ADCSRA.ADEN=1, AIN1 se aplica a la entrada negativa del comparador analógico.

Table 28-1 Analog Comparator Multiplexed Input

ACME	ADEN	MUX[2..0]	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

(/local--files/8avr:analog-comparator-voltage-reference/analog-comparator-mux-input.png)

## Control de comparador analógico y registro de estado B

El registro de control y estado de la memoria del programa de almacenamiento contiene los bits de control necesarios para controlar las operaciones del cargador de arranque. Al direccionar registros de E/S como espacio de datos usando instrucciones LD y ST, se debe usar el desplazamiento proporcionado. Cuando se utilizan los comandos IN y OUT específicos de E/S, el desplazamiento se reduce en 0x20, lo que da como resultado un desplazamiento de la dirección de E/S entre 0x00 y 0x3F.

Nombre: ACSRB Compensación: 0x4F Restablecimiento: 0x00

Propiedad:

Cuando se direcciona como registro de E/S: la compensación de dirección es 0x2F

Bit	7	6	5	4	3	2	1	0
Access								ACOE
Reset								R/W 0

(/local--files/8avr:analog-comparator-voltage-reference/input-register.png)

- Bit 0 – ACOE: Habilitación de salida de comparador analógico

Cuando se establece este bit, la salida del comparador analógico se conecta al pin ACO.

## Registro de control y estado del comparador analógico

Al direccionar registros de E/S como espacio de datos usando instrucciones LD y ST, se debe usar el desplazamiento proporcionado. Cuando se utilizan los comandos IN y OUT específicos de E/S, el desplazamiento se reduce en 0x20, lo que da como resultado un desplazamiento de la dirección de E/S entre 0x00 y 0x3F.

Nombre: ACSR Compensación: 0x50 Restablecimiento: N/A

Propiedad:

Cuando se direcciona como registro de E/S: la compensación de dirección es 0x30

Bit	7	6	5	4	3	2	1	0
Access	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACISO
Reset	R/W 0	R/W 0	R a	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0

(/local--files/8avr:analog-comparator-voltage-reference/status-register.png)

- Bit 7 – ACD: Deshabilitar comparador analógico

Cuando este bit se escribe como uno lógico, se desconecta la alimentación del comparador analógico. Este bit se puede configurar en cualquier momento para apagar el comparador analógico. Esto reducirá el consumo de energía en los modos Activo e Inactivo. Al cambiar el bit ACD, la interrupción del comparador analógico debe desactivarse borrando el bit ACIE en ACSR. De lo contrario, puede ocurrir una interrupción cuando se cambia el bit.

- Bit 6 – ACBG: Selección de intervalo de banda del comparador analógico

Cuando se establece este bit, un voltaje de referencia de banda prohibida fija reemplaza la entrada positiva al comparador analógico. Cuando se borra este bit, se aplica AIN0 a la entrada positiva del comparador analógico. Cuando la referencia de banda prohibida se utiliza como entrada al comparador analógico, el voltaje tardará cierto tiempo en estabilizarse. Si no se estabiliza, la primera conversión puede dar un valor incorrecto.

- Bit 5 – ACO: Salida del comparador analógico

La salida del comparador analógico se sincroniza y luego se conecta directamente a ACO. La sincronización introduce un retraso de uno a dos ciclos de reloj.

- Bit 4 – ACI: Indicador de interrupción del comparador analógico

Este bit lo establece el hardware cuando un evento de salida del comparador activa el modo de interrupción definido por ACIS1 y ACIS0. La rutina ACI se ejecuta si se establece el bit ACIE y se establece el bit I en SREG. El hardware borra ACI cuando se ejecuta el vector de manejo de interrupción correspondiente. Alternativamente, ACI se borra escribiendo un uno lógico en la bandera.

- Bit 3 – ACIE: Habilitación de interrupción del comparador analógico

Cuando el bit ACIE se escribe uno lógico y se establece el bit I en el registro de estado, se activa la interrupción del comparador analógico. Cuando se escribe cero lógico, la interrupción se deshabilita.

- Bit 2 – ACIC: Habilitación de captura de entrada de comparador analógico

Cuando se escribe como uno lógico, este bit permite que el comparador analógico active la función de captura de entrada en el temporizador/contador 1. En este caso, la salida del comparador está directamente conectada a la lógica frontal de captura de entrada, lo que hace que el comparador utilice las funciones de cancelación de ruido y selección de borde de la interrupción de captura de entrada del temporizador/contador1. Cuando se escribe cero lógico, no existe conexión entre el comparador analógico y la función de captura de entrada. Para hacer que el comparador dispare la interrupción de captura de entrada del temporizador/contador 1, se debe establecer el bit ICIE1 en el registro de máscara de interrupción del temporizador (TIMSK1).

- Bits 1:0 – ACISn: selección del modo de interrupción del comparador analógico [n = 1:0]

Estos bits determinan qué eventos del comparador activan la interrupción del comparador analógico.

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

(/local--files/8avr:analog-comparator-voltage-reference/comparator-interruptut.png)

Al cambiar los bits ACIS1/ACIS0, la interrupción del comparador analógico debe desactivarse borrando su bit de habilitación de interrupción en el registro ACSR. De lo contrario, puede ocurrir una interrupción cuando se cambian los bits.

## Registro de desactivación de entrada digital 1

Nombre: DIDR1 Compensación: 0x7F ; Restablecer: 0x00

Propiedad:

Bit	7	6	5	4	3	2	1	0
Access							AIN1D	AIN0D
Reset							R/W	R/W

(/local--files/8avr:analog-comparator-voltage-reference/digital-input-disable.png)

- Bit 1 – AIN1D: Desactivación de entrada digital AIN1
- Bit 0 – AIN0D: AIN0 Inhabilitación de entrada digital

Cuando este bit se escribe uno lógico, el búfer de entrada digital en el pin AIN1/0 está deshabilitado. El bit de registro de PIN correspondiente siempre se leerá como cero cuando se establezca este bit. Cuando se aplica una señal analógica al pin AIN1/0 y no se necesita la entrada digital de este pin, este bit debe escribirse uno lógico para reducir el consumo de energía en el búfer de entrada digital.

Especificación de voltaje de referencia ADC de 1,1 V seleccionable:

Symbol	Parameter	Condition	Min.	Typ	Max	Units
$V_{POT}$	Power-on Reset Threshold Voltage (rising)		1.1	1.5	1.7	V
	Power-on Reset Threshold Voltage (falling) <sup>(2)</sup>		0.6	1.0	1.7	V
$S_{R_{ON}}$	Power-on Slope Rate		0.01	-	10	V/ms
$V_{RST}$	RESET Pin Threshold Voltage		0.2 $V_{CC}$	-	0.9 $V_{CC}$	V
$t_{RST}$	Minimum pulse width on RESET Pin		-	-	2.5	$\mu$ s
$V_{HYST}$	Brown-out Detector Hysteresis		-	50	-	mV
$t_{BOD}$	Min. Pulse Width on Brown-out Reset		-	2	-	$\mu$ s
$V_{BG}$	Bandgap reference voltage	$V_{CC}=2.7$ $T_A=25^\circ C$	1.0	1.1	1.2	V
$t_{BG}$	Bandgap reference start-up time	$V_{CC}=2.7$ $T_A=25^\circ C$	-	40	70	$\mu$ s
$I_{BG}$	Bandgap reference current consumption	$V_{CC}=2.7$ $T_A=25^\circ C$	-	10	-	$\mu$ A

(/local--files/8avr:analog-comparator-voltage-reference/adc-reference.png)

## Proyecto de ejemplo

Un ejemplo de proyecto para el comparador analógico está disponible aquí: Proyecto de ejemplo (/8avr:analog-comparator-voltage-reference)

# 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 y 16 bits de ancho y ofrecen diferentes funciones según 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:

Temporizador 0	TC0	Temporizador/contador de 8 bits con modulación de ancho de pulso (PWM)
Temporizador 1	TC1	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono
Temporizador 2	TC2	Temporizador/contador de 8 bits con PWM y funcionamiento asíncrono
Temporizador 3	TC3	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono
Temporizador 4	TC4	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono

## Definiciones:

### Nomenclatura de registro

ABAJO	El contador llega al FONDO cuando llega a cero (0x00 para contadores de 8 bits y 0x0000 para contadores de 16 bits)
MÁX.	El contador alcanza su valor máximo cuando pasa a ser 0x0F (15 decimal) para contadores de 8 bits y 0x00FF (255 decimal) para contadores de 16 bits
PARTE SUPERIOR	El contador llega al TOP cuando su valor llega a ser 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 uso general de 8 bits, con dos unidades de comparación de salida independientes y compatibilidad con PWM.

## Registros TC0

El registro del temporizador/contador 0 (TCNT0) y los registros de comparación de salida TC0x (OCR0x) son registros de 8 bits. Todas las señales de solicitud de interrupción son visibles en el registro de bandera 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

(/local--files/8avr:avrtimer/TCCR0B.PNG)

## Fuentes de reloj del temporizador/contador TC0

TC0 puede sincronizarse con 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 control de temporizador/contador (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	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

## Unidad de contador TC0

Dependiendo del modo de operación utilizado, T0 se borra, aumenta o disminuye en cada reloj del temporizador (clkT0). clkT0 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (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 funcionamiento admitidos por la unidad de temporizador/contador son: modo normal (contador), modo Borrar temporizador en comparación (CTC) y dos tipos de modos de modulación de ancho de pulso (PWM).

Tabla 1-2 Bit de modo de generación de forma de onda Descripción

Modo	WGM2:0	Modo de operación	PARTE SUPERIOR	Actualización OCR0x	Indicador TOV activado
0	0 0 0	Normal	0xFF	Inmediato	MÁX.
1	0 0 1	Fase PWM correcta	0xFF	PARTE SUPERIOR	ABAJO
2	0 1 0	CTC	OCRA	Inmediato	MÁX.
3	0 1 1	PWM rápido	0xFF	ABAJO	MÁX.
4	1 0 0	Reservado	~	~	~
5	1 0 1	Fase PWM correcta	OCRA	PARTE SUPERIOR	ABAJO
6	1 1 0	Reservado	~	~	~
7	1 1 1	PWM rápido	OCRA	ABAJO	MÁX.

Nota:

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

## Modos de operación para TC0

El modo de operación determina el comportamiento de TC0 y los pines de comparación de salida. Se define mediante la combinación de los bits del modo de generación de forma de onda y los bits del modo de salida de comparación en los registros de control del temporizador/contador A y B (TCCR0B.WGMn2, TCCR0A.WGM01, TCCR0A.WGM00 y TCCR0A.COM0x[1:0]).

Los modos de operación disponibles para TC0 son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

### **Borrar temporizador en el modo de comparación de coincidencias**

En el modo Clear Timer on Compare o CTC (WGM0[2:0]=0x2), el **registro OCR0A** se usa 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 máximo del contador y, por lo tanto, 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 alcance el valor SUPERIOR configurando el indicador OCF0A. Si la interrupción está habilitada, la rutina del controlador de interrupciones se puede usar para actualizar el valor SUPERIOR.

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

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

(/local--files/8avr:avrtimer/freqz0.PNG)

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

## **TC1, TC3 y TC4: temporizador/contador de 16 bits con PWM**

Las unidades de temporizador/contador de 16 bits permiten la sincronización precisa de la ejecución del programa (gestión de eventos), la generación de ondas y la medición de la sincronización de la señal.

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

- El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son todos registros de 16 bits.
- Los registros de control de temporizador/contador (TCCRnA/B) son registros de 8 bits y no tienen restricciones de acceso a la CPU.
- 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 máscara de interrupción del temporizador (TIMSKn).

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

El temporizador/contador puede sincronizarse con una fuente de reloj interna o externa. La fuente de reloj se selecciona mediante la lógica de selección de reloj, que está controlada por los bits de selección de reloj en el registro B de control de temporizador/contador (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
Access	ICNC	ICES		WGM3	WGM2		CS[2:0]	
Reset	0	0		0	0	0	0	0

(/local--files/8avr:avrtimer/TCCR1B.PNG)

Bits 2:0 – CS[2:0]: 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	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

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

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

Cada contador de 16 bits se asigna a dos ubicaciones de memoria de E/S de 8 bits: **Contador alto** (TCNTnH) que contiene los ocho bits superiores del contador y **Contador bajo** (TCNTnL) que contiene los ocho bits inferiores.

Dependiendo del modo de operación seleccionado, el contador se borra, incrementa o decremente en cada reloj del temporizador (clkTn). El reloj clkTn se puede generar a partir de una fuente de reloj externa o interna, según lo seleccionado 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 control de temporizador/contador A y B (TCCRnB.WGM[3:2] y TCCRnA.WGM[1:0]).

## Modos de operación (TC1, TC3, TC4)

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

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase
- Modo PWM de corrección 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 pendiente única. El contador cuenta de ABAJO a ARRIBA y luego se reinicia desde ABAJO.

En el modo Fast PWM, 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). A continuación, 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:

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

(/local--files/8avr:avrtimer/freqz1.PNG)

Nota:

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

## TC2 - Timer/Counter2 de 8 bits con PWM y funcionamiento asíncrono

Timer/Counter2 (TC2) es un módulo de temporizador/contador de 8 bits de doble canal y propósito general.

### Registros TC2

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

**Name:** TCCR2B  
**Offset:** 0xB1  
**Reset:** 0x00  
**Property:** -

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

(/local--files/8avr:avrtimer/TCCR2B.PNG)

### Fuentes de reloj TC2

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

#### CS02 CS01 CS00 Descripción

0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)

0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

## Unidad de contador TC2

Dependiendo del modo de operación utilizado, el contador se borra, incrementa o decrementa en cada reloj del temporizador (clkT2). clkT2 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (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 del modo de generación de forma de onda (WGM2[2:0]) y el modo de salida de comparación (COM2x[1:0]) pedacitos

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

### Modo normal

En el modo Normal (WGM22:0 = 0) la dirección de conteo siempre es hacia arriba (incrementando) sin que se borre el contador. El contador cambiará a 0c00 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 cuando el TCNT2 se vuelve cero. El indicador TOV2, en este caso, se comporta como un noveno bit, excepto que solo se establece, no se borra. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el

indicador TOV2, el software puede aumentar la resolución del temporizador. 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 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

- ATmega328PB Mini tablero explicado
- Cable micro USB

#### Herramienta



(<http://www.atmel.com/tools/MEGA328PB-XMINI.aspx>)

Mini kit de evaluación

**ATmega328PB Xplained**

#### Sobre



(<http://www.atmel.com/tools/MEGA328PB-XMINI.aspx>)



(<https://www.microchipdirect.co>)

El kit de evaluación ATmega328PB Xplained Mini es una plataforma de hardware para evaluar el microcontrolador Atmel ATmega328PB. NO se necesita un kit de depurador externo para ejecutar estos laboratorios. El ATmega328PB tiene un depurador incorporado completamente integrado.

Este ejercicio práctico demuestra cómo configurar los diferentes temporizadores.

### Herramientas de software

#### Instaladores



Windows



linux



Mac OS X

! Instrucciones de instalación

#### Herramienta

#### Sobre

Entorno de desarrollo integrado  
Atmel® Studio



([/atstudio:start](#))



(<http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe>)



([/install:atstudi](#))



([/install:atstudi](#))

### Archivos adicionales

#### archivos



Guía del usuario del tablero explicado ([http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide\\_UserGuide.pdf](http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf))

## Descripción general de los temporizadores en MCU AVR de 8 bits

El microcontrolador ATmega328PB tiene cinco temporizadores/contadores:

Temporizador 0	TC0	Temporizador/contador de 8 bits con PWM
Temporizador 1	TC1	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.
Temporizador 2	TC2	Temporizador/contador de 8 bits con PWM y funcionamiento asíncrono.
Temporizador 3	TC3	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.
Temporizador 4	TC4	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.

#### Definiciones:

### Nomenclatura de registro

ABAJO	El contador llega al FONDO cuando llega a cero (0x00 para contadores de 8 bits y 0x0000 para contadores de 16 bits)
MÁX.	El contador alcanza su valor máximo cuando pasa a ser 0x0F (15 decimal) para contadores de 8 bits y 0x00FF (255 decimal) para contadores de 16 bits
PARTE SUPERIOR	El contador llega al TOP cuando su valor llega a ser igual al valor más alto posible. Al valor TOP se le puede asignar el valor fijo MAX o el valor almacenado en el registro OCRxA. Esta asignación depende del modo de funcionamiento

## Temporizador 0 - Temporizador/Contador de 8 bits con PWM

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

### Registros TC0

El registro del temporizador/contador 0 (TCNT0) y los registros de comparación de salida TC0x (OCR0x) son registros de 8 bits. Todas las señales de solicitud de interrupción son visibles en el registro de bandera 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

(/local--files/8avr:avrtimer/TCCR0B.PNG)

### Fuentes de reloj del temporizador/contador TC0

TC0 puede sincronizarse con 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 control de temporizador/contador (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	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

### Unidad de contador TC0

Dependiendo del modo de operación utilizado, T0 se borra, aumenta o disminuye en cada reloj del temporizador (clkT0). clkT0 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (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 Borrar temporizador en comparación (CTC) y dos tipos de modos de modulación de ancho de pulso (PWM).

Tabla 1-2 Bit de modo de generación de forma de onda Descripción

Modo	WGM2:0	Modo de operación	PARTE SUPERIOR	Actualización OCR0x	Indicador TOV activado
0	0 0 0	Normal	0xFF	Inmediato	MÁX.
1	0 0 1	Fase PWM correcta	0xFF	PARTE SUPERIOR	ABAJO
2	0 1 0	CTC	OCRA	Inmediato	MÁX.
3	0 1 1	PWM rápido	0xFF	ABAJO	MÁX.
4	1 0 0	Reservado	~	~	~
5	1 0 1	Fase PWM correcta	OCRA	PARTE SUPERIOR	ABAJO
6	1 1 0	Reservado	~	~	~
7	1 1 1	PWM rápido	OCRA	ABAJO	MÁX.

**Nota:**

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

**Modos de operación para TC0**

El modo de operación determina el comportamiento de TC0 y los pines de comparación de salida. Se define mediante la combinación de los bits del modo de generación de forma de onda y los bits del modo de salida de comparación en los registros de control del temporizador/contador A y B (TCCR0B.WGMn2, TCCR0A.WGM01, TCCR0A.WGM00 y TCCR0A.COM0x[1:0]).

Los modos de operación disponibles para TC0 son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

**Borrar temporizador en el modo de comparación de coincidencias**

En el modo Clear Timer on Compare (WGM0[2:0]=0x2), el registro OCR0A se usa 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 máximo del contador y, por lo tanto, 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 alcance el valor SUPERIOR configurando el indicador OCF0A. Si la interrupción está habilitada, la rutina del controlador de interrupciones se puede usar para actualizar el valor SUPERIOR.

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

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

(/local--files/8avr:avrtimer/freqz0.PNG)

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

**TC1, TC3 y TC4: temporizador/contador de 16 bits con PWM**

Las unidades de temporizador/contador de 16 bits permiten la sincronización precisa de la ejecución del programa (gestión de eventos), la generación de ondas y la medición de la sincronización de la señal.

**Registros (TC1, TC3, TC4)**

- El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son todos registros de 16 bits.
- Los registros de control de temporizador/contador (TCCRnA/B) son registros de 8 bits y no tienen restricciones de acceso a la CPU.
- 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 máscara de interrupción del temporizador (TIMSKn).

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

El temporizador/contador puede sincronizarse con una fuente de reloj interna o externa. La fuente de reloj se selecciona mediante la lógica de selección de reloj, que está controlada por los bits de selección de reloj en el registro B de control de temporizador/contador (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
Access	ICNC	ICES		WGM3	WGM2		CS[2:0]	
Reset	0	0		0	0	0	0	0

(/local--files/8avr:avrtimer/TCCR1B.PNG)

Bits 2:0 – CS[2:0]: 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	Descripción
0	0	0	Sin fuente de reloj (Temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

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

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

Cada contador de 16 bits se asigna a dos ubicaciones de memoria de E/S de 8 bits: Contador alto (TCNTnH) que contiene los ocho bits superiores del contador y Contador bajo (TCNTnL) que contiene los ocho bits inferiores.

Dependiendo del modo de operación seleccionado, el contador se borra, incrementa o decremente en cada reloj del temporizador (clkTn). El reloj clkTn se puede generar a partir de una fuente de reloj externa o interna, según lo seleccionado 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 control de temporizador/contador A y B (TCCRnB.WGM[3:0] y TCCRnA.WGM[1:0]).

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

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

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase
- Modo PWM de corrección 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 pendiente única. El contador cuenta de ABAJO a ARRIBA y luego se reinicia desde ABAJO.

En el modo Fast PWM, 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). A continuación, 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:

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

(/local--files/8avr:avrtimer/freqz1.PNG)



#### Nota:

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

## TC2 - Timer/Counter2 de 8 bits con PWM y funcionamiento asíncrono

Timer/Counter2 (TC2) es un módulo de temporizador/contador de 8 bits de doble canal y propósito general.

### Registros TC2

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

Name: TCCR2B  
Offset: 0xB1  
Reset: 0x00  
Property: -

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

(/local--files/8avr:avrtimer/TCCR2B.PNG)

### Fuentes de reloj TC2

TC2 puede sincronizarse con una fuente de reloj síncrona interna o asíncrona externa:

Los tres bits de selección de reloj (CS2:CS0) seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

### Unidad de contador TC2

Dependiendo del modo de operación utilizado, el contador se borra, incrementa o decrementa en cada reloj del temporizador (clkT2). clkT2 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (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 del modo de generación de forma de onda (WGM2[2:0]) y el modo de salida de comparación (COM2x[1:0]) pedacitos

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

#### Modo normal

En el modo Normal (WGM22:0 = 0) el sentido de conteo es siempre ascendente (incremental), sin que se borre el contador. El contador cambiará a 0c00 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 cuando el TCNT2 se vuelve cero. El indicador TOV2, en este caso, se comporta como un noveno bit, excepto que solo se establece, no se borra. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el indicador TOV2, el software puede aumentar la resolución del temporizador. No hay casos especiales a considerar en el modo Normal, se puede escribir un nuevo valor de contador en cualquier momento.



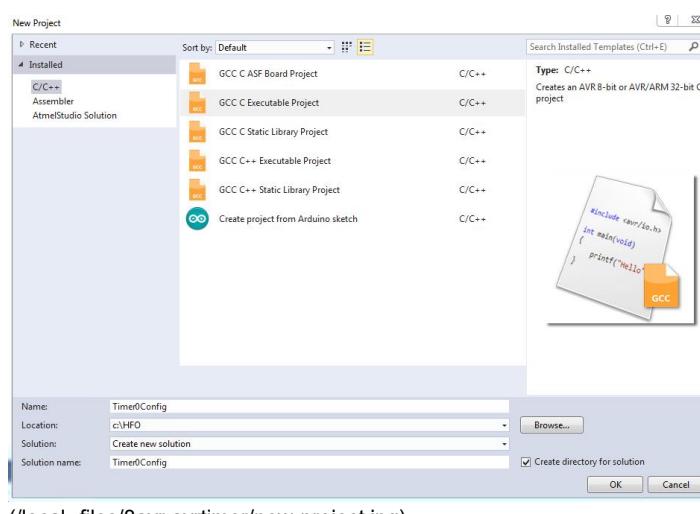
Independientemente del modo que se utilice, el programador debe recordar dos cosas:

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

## ! Paso a paso

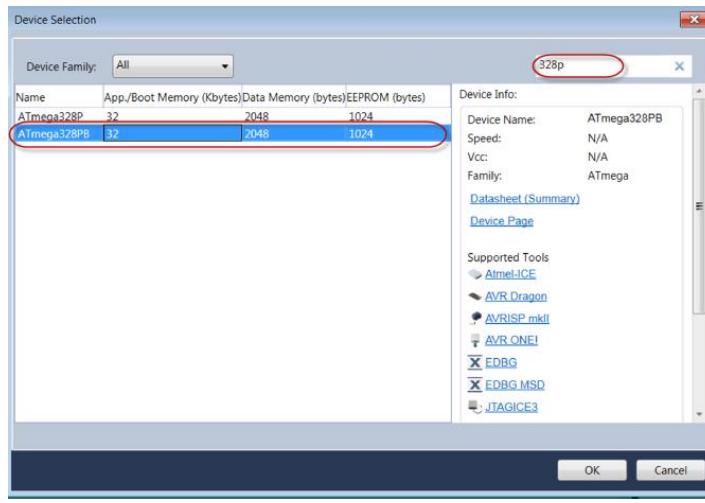
### Creación de proyectos

- 1 Abrir Atmel Studio 7
- 2 Seleccione **Archivo > Nuevo > Proyecto**
- 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 del proyecto. (Este ejemplo usa "C:\HFO\TimersConfigurations"). Haga clic en **Aceptar** .



(/local--files/8avr:avrTimer/new-project.jpg)

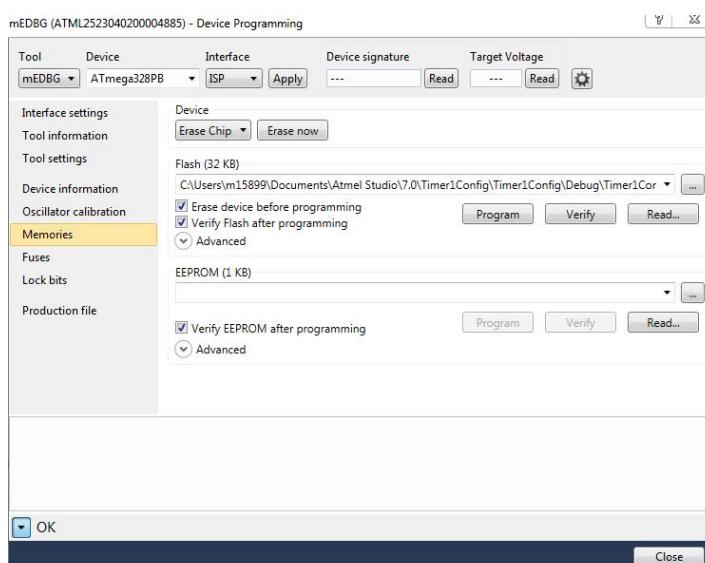
- 4 En la barra de búsqueda de la ventana **Selección de dispositivo** , escriba "328p", luego seleccione el dispositivo **Atmega328PB** , haga clic en **Aceptar** .



(/local--files/8avr:avrtimer/select-device.jpg)

## Programando la placa

- 5 Seleccione **Build > Build Solution** 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 se muestra en la imagen a continuación y haga clic en **Aplicar**.



(/local--files/8avr:avrtimer/program.jpg)

- 7 Selecciona **Memorias > Programa** Espera hasta que aparezca el mensaje "Verificando Flash...OK".

## Modificaciones del proyecto



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

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

## 8 Modificación 1: parpadeo de un LED usando TC0

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

- Modifique la función principal agregando el siguiente código:



```

1 int main(void) ?
2 {
3     //set RB5 as output
4     DDRB |= 1 << DDRB5;
5
6     //call TMR0 initial
7     init_TC0();
8
9     //enable interrupt
10    sei();
11
12    while (1)
13    {
14        //main loop
15    }
16    dieciséis
17 }
```

**b** Cree la función `init_TC0()` en `main.c`. Agrega el siguiente código:

```

1 void init_TC0(void) ?
2 {
3     // Set the Timer Mode to CTC
4     TCCR0A |= (1 << WGM01);
5
6     // Set the value that you
7     OCR0A = 0xF9; //249
8
9     //Set the ISR COMPA vect
10    TIMSK0 |= (1 << OCIE0A);
11
12    // set prescaler to 1024
13    TCCR0B |= (1 << CS00) | (1
14
15 }
```

El código anterior hace lo siguiente:

- Configura el Timer 0 en modo CTC configurando el bit WGM correspondiente en el registro TCCR0A;
- Establece el valor TOP en el registro OCR0A calculado usando la ecuación 1 para hacer parpadear el LED a 30 kHz;
- Habilita la interrupción del temporizador;
- Establece el escalador previo a 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
4     counter++;
5     if (counter == MyTimerConst)
6         counter = 0;
7     PORTB ^= 1 << PORTB5;
8 }
9 }
```



El uso de la constante `MyTimerConstant` y la variable `contador` 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

Este ejemplo usa TC1 para generar una señal PWM que se alimenta a través de un pin de E/S para controlar un LED. Cambiar el ciclo de trabajo de PWM dará como resultado un cambio en el brillo del LED.

**a** Modifique `main()` agregando el siguiente código.



```

1 int main(void) ?▲
2 {
3     //set direction of pins
4     DDRB |= 1 << DDRE;
5
6     init_TC1_pwm();
7
8     //enable global interrupts
9     sei();
10
11    while (1)
12    {
13        //main loop
14    }
15
16 dieciséis } ▾

```

Para configurar el bit de dirección del pin PB1, escriba el bit DDB1 en 1 lógico en el registro DDRB. En la miniplaca ATmega328PB Xplained, 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 miniplaca ATmega328PB Xplained, como se muestra a continuación.

- b** Crea la función init\_TC1\_pwm() antes del bucle principal con el siguiente código:

```

1 void init_TC1_pwm(void) ?▲
2 {
3     //clear OCnA on compare match
4     TCCR1A = (1 << COM1A1);
5
6     //the counting sequence is
7     TCCR1A |= (1 << WGM10);
8     TCCR1B |= (1 << WGM12);
9
10    //256 prescaler clock selection
11    TCCR1B |= (0 << CS10) | (1 << CS11);
12
13    //enable interrupts
14    TIMSK0 |= (1 << OCIE1A);
15 }

```

init\_TC1\_PWM() realiza lo siguiente"

- Establece los bits COMA1=1 y COMA0=0 (modo no inversor de la tabla de la hoja de datos Modo de salida de comparación, PWM rápido en el registro TCCR1A);
- Configura Bits 1:0 – WGM[1:0]:Modo de generación de forma de onda en consecuencia para Fast PWM, modo de 8 bits;
- 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;
- 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 }

```

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

## 10 Parpadeo de un LED usando TC2

Para este ejemplo, el temporizador 2 (TC2) está configurado para hacer 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
4     DDRB |= 1 << DDB5;
5
6     /* Timer clock = 1,
7      TCCR2B = (1 << CS2);
8
9     /* Clear overflow 1
10    TIFR2 = 1 << TOV2;
11
12    /* Enable Overflow
13    TIMSK2 = 1 << TOIE;
14
15    // enable global ir
16    sei();
17
18    while (1)
19    {
20        // Main loop
21    }
22 }
```

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

**b** Habilite la interrupción del temporizador de desbordamiento;

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

```

1 ISR (TIMER0_COMPA_vect)      ?
2 {
3     counterTimer2++;
4     if(counterTimer2 == MyTim
5     {
6         counterTimer2 = 0;
7         PORTB ^= 1 << PORTB5;
8     }

```

Configure el registro PORTB para que parpadee el LED



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

# Ejemplo de sensor de temperatura interna del convertidor analógico a digital (ADC) AVR de 8 bits

## ⌚ Objetivo

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

El sensor de temperatura en el chip está acoplado a un canal ADC8 de un solo extremo. Seleccionar el 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 usar en modo de conversión simple 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 ±10°C.

## ☒ Materiales

### Herramientas de hardware ( *opcional* )

Herramienta	Sobre
 (http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	Mini kit de evaluación <b>ATmega328PB-Xplained</b> <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">?</a> (http://www.atmel.com/tools/MEGA328PB-XMINI.aspx) <a href="https://www.microchipdirect.com/ATMEGA328PB-XMINI.aspx">!</a> (https://www.microchipdirect.com/ATMEGA328PB-XMINI.aspx)

### Herramientas de software

Herramienta	Sobre	Instaladores				Instrucciones de instalación
		Windows	Linux	Mac OS X		
Entorno de desarrollo integrado <b>Atmel® Studio</b>	<a href="http://atstudio:start">?</a> (/atstudio:start) <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">!</a> (http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe)	<a href="http://atstudio:start">!</a> (http://atstudio:start)	<a href="http://atstudio:linux">!</a> (http://atstudio:linux)	<a href="http://atstudio:mac">!</a> (http://atstudio:mac)	<a href="http://atstudio:install">!</a> (/install:atstudio)	

### Archivos de ejercicios

Expediente	Windows
 Archivo fuente Main.c	<a href="https://microchiptechnology.sharepoint.com/:f/s/DeveloperHelp/EgvbV6I04yBAvXfl6BemPdoBdqWAfKoruX9pkt21RgFG-w?e=pCb004">!</a> (https://microchiptechnology.sharepoint.com/:f/s/DeveloperHelp/EgvbV6I04yBAvXfl6BemPdoBdqWAfKoruX9pkt21RgFG-w?e=pCb004)

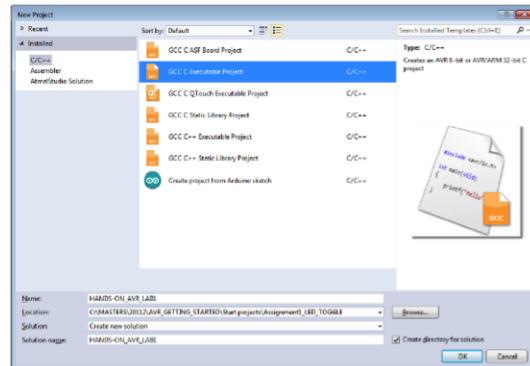
### Archivos adicionales

archivos
 <a href="http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf">Guía del usuario del tablero explicado</a> (http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf)

## Procedimiento

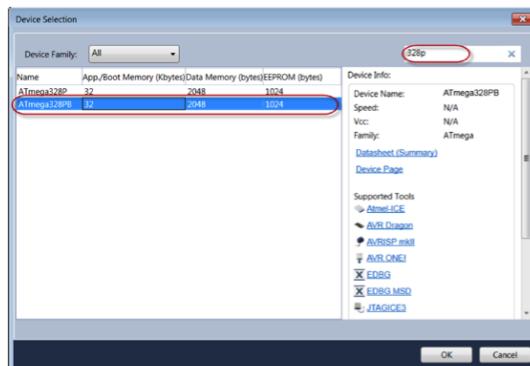
### 1 Tarea 1: creación de proyectos

- Abrir Atmel Studio 7
- Seleccione **Archivo > Nuevo > Proyecto**
- Seleccione Proyecto ejecutable GCC C y asígnele el nombre **Proyecto1**
- Elija una ubicación para guardar el proyecto en su computadora



(/local--files/8avr:avradc/step1.png)

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



(/local--files/8avr:avradc/step2.png)

### 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 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)
        ;

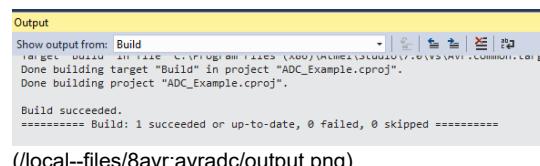
        /* Calculate the temperature in C */
        Ctemp = (ADC - 247)/1.22;
        Ftemp = (Ctemp * 1.8) + 32;
    }

    return -1;
}

```

### 3 Tarea 3 - Proyecto de construcción

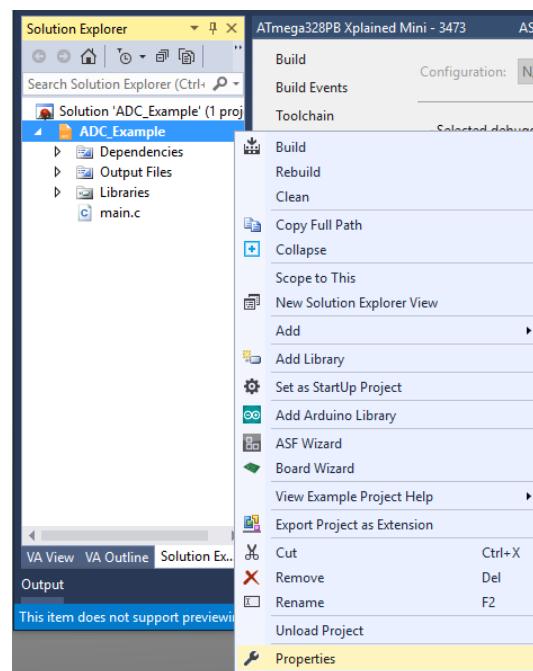
- Seleccione **Build > Build Solution** en el menú de Studio 7 para compilar el código. Verá un mensaje de **Build Succeeded** en la ventana de resultados. Si hay algún error, verifique `main.c` para ver si hay errores al ingresar el código del programa.



(/local--files/8avr:avradc/output.png)

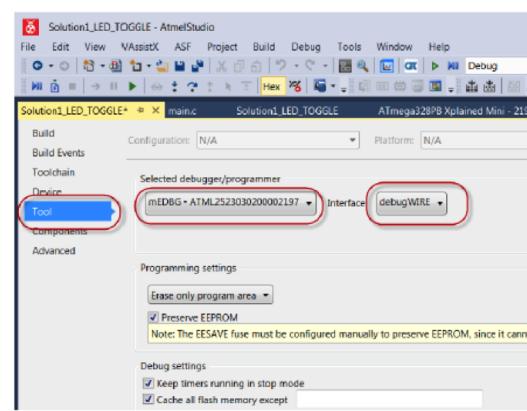
### 4 Tarea 4: programación de la placa explicada

- Conecte la placa Xplained al puerto USB de la computadora usando el cable incluido.
- En el área **Explorador de soluciones**, haga clic con el botón derecho en el nombre del proyecto y seleccione **Propiedades**.



(/local--files/8avr:avradc/solution.png)

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



(/local--files/8avr:avradc/step42.png)

- Seleccione **Depurar > Iniciar sin depurar** en el menú de Studio 7. El proyecto construirá y luego programará el tablero explicado con el código del proyecto junto con el control de depuración.



(/local--files/8avr:avradc/step52.png)

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

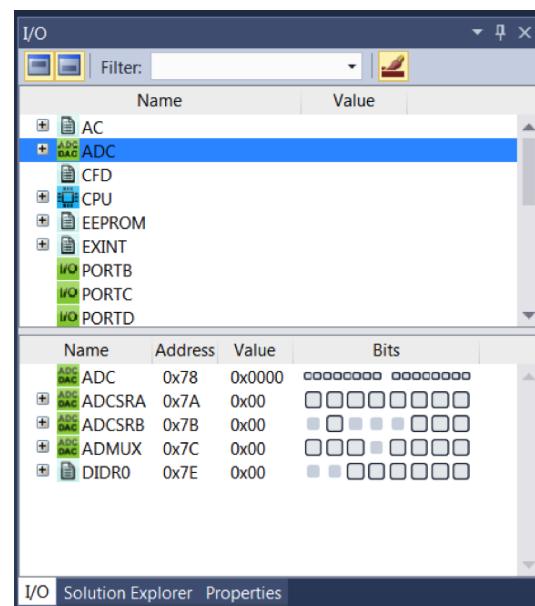


Si la placa explicada no se conecta, es posible que haya un ajuste de fusible que provoque que esto ocurra (/boards:debugbrick).

## 5 Tarea 5 - Depuración

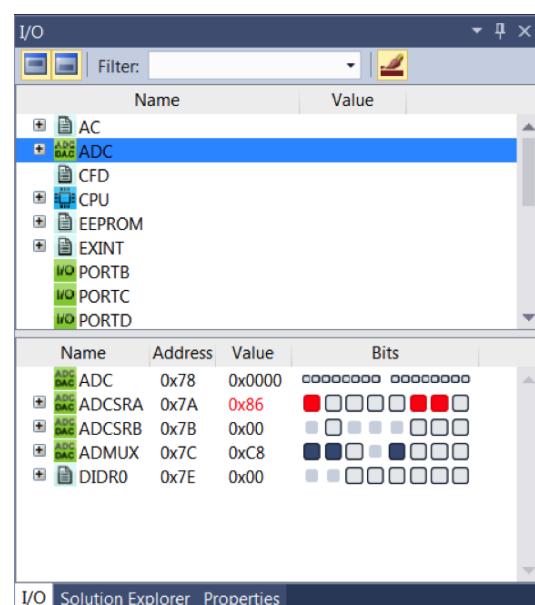
La depuración de un dispositivo es esencial para determinar cómo se puede ejecutar un programa.

- Seleccione **Depurar > Iniciar depuración y romper** . El proyecto se compilará y el programa se cargará en el tablero Xplained
- Se abrirá la ventana I/O View mostrando los diversos periféricos
- Haga clic en la selección de **ADC** para abrir la vista de E/S para el ADC



(/local--files/8avr:avradc/adcdebug.png)

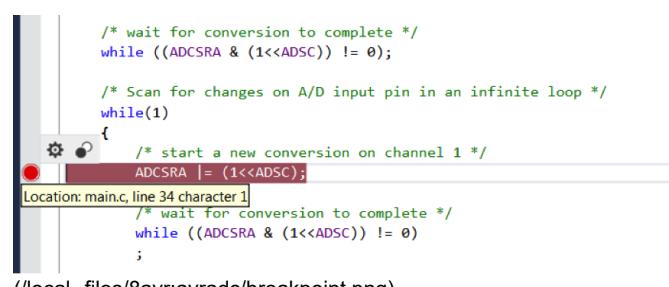
- Seleccione **Depurar > Pasar por alto (F10)** para recorrer el programa en un solo paso en el tablero Xplained. Supervise los registros ADC en la vista de E/S mientras ejecuta un solo paso



(/local--files/8avr:avradc/adcdebug2.png)

## 6 Tarea 6: punto de interrupción y salida

- Haga clic en **Depurar > Romper todo** en el menú superior de Studio 7
- Haga clic en el margen para habilitar un punto de interrupción en la línea de comando en la instrucción ADCSRA dentro del ciclo While



(/local--files/8avr:avradc/breakpoint.png)

- 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 él

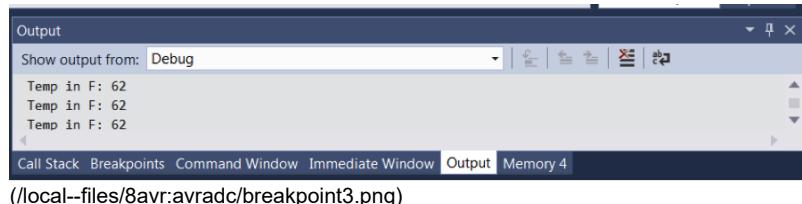


(/local--files/8avr:avradc/breakpoint2.png)

- Dentro de la **Ventana** de configuración de punto de interrupción , marque la casilla **Acciones** . Dentro de "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**
- Ingrese al modo de depuración haciendo clic en **Iniciar depuración y romper** en el menú de **depuración superior**
- Abra la **ventana de salida** seleccionando **Depurar > Ventanas > Salida** para ver el valor de la variable de temperatura
- Haga clic en **Depurar > Continuar** y vea la temperatura en la ventana Salida

## Resultados

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



(/local--files/8avr:avradc/breakpoint3.png)

### 7 Tarea 7: deshabilitar debugWIRE y cerrar

El **fusible debugWire** debe restablecerse para poder 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.

## Q Análisis

Usar el ADC es bastante fácil y la función de depuración facilita el seguimiento de los resultados.

## 💡 Conclusiones

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

# Sensor de temperatura interna 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 el chip que está acoplado a un canal ADC de un solo extremo. Seleccionar el canal ADC 8 escribiendo '1000' en ADMUX.MUX[3:0] habilita el sensor de temperatura. La referencia de voltaje interno 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 usar en modo de conversión simple para medir el voltaje sobre el sensor de temperatura.

## Muestra de datos de medición

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

(/local--files/8avr:avrtemp/temp1.png)

## Calibración

Los resultados de las mediciones de temperatura tienen errores de compensación y ganancia. La referencia de temperatura interna puede corregirse 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.  
([http://www.atmel.com/Images/Atmel-8108-Calibration-of-the-AVR's-Internal-Temperature-Reference\\_ApplicationNote\\_AVR122.pdf](http://www.atmel.com/Images/Atmel-8108-Calibration-of-the-AVR's-Internal-Temperature-Reference_ApplicationNote_AVR122.pdf))

# Configuración del ADC

La referencia de voltaje interno de 1,1 V debe seleccionarse como fuente de referencia de voltaje ADC cuando se usa el sensor de temperatura interno. Escribir "11" en los bits **REFS1** y **REFS0** del **registro ADMUX** 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 simple** se puede usar para convertir la señal del sensor de temperatura conectado al canal 8. Para seleccionar el canal 8, escribir "1000" en los bits MUX3 a MUX0 selecciona el canal 8 o el sensor de temperatura.

Una vez que se completa la conversión, el resultado se almacena en dos registros de datos ADC de 8 bits ADCH (8 bits superiores) y ADCL (8 bits inferiores). El resultado de 10 bits puede estar justificado a la izquierda o la derecha. Si el bit ADLAR se establece en "1", 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 se borra y la palabra se justifica a la derecha.

Esta declaración de código establecerá los bits como se describe.

ADMUX = (1«REF1) | (1«REF0) | (0«ADLR) | (1«MUX3) | (0«MUX2) | (0«MUX1) | (0«MUX0);

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

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

#### 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

(/local--files/8avr:avrtemp/temp2.png)

**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

(/local--files/8avr:avrtemp/temp3.png)

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 <sub>BG</sub> )
1111	0V (GND)

(/local--files/8avr:avrtemp/temp4.png)

## Configuración del reloj ADC y el tiempo de conversión

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

El valor del preescalador se selecciona con **bits ADPS** en **el registro ADCSRA**. Por ejemplo; escribir "110" en **el registro ADCSRA** selecciona el preescalador de división por 64 que da como resultado un reloj ADC de 125 KHz cuando se usa un reloj oscilador de 8 MHz.

Registro A de control y estado del ADC

**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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(/local--files/8avr:avrtemp/temp5.png)

**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.

(/local--files/8avr:avrtemp/temp6.png)

**Table 29-5 Input Channel Selection**

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

(/local--files/8avr:avrtemp/temp7.png)

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

(/local--files/8avr:avrtemp/temp8.png)

## Iniciar una conversión

En el modo de conversión simple, el **bit ADSC** en el **registro ADCSRA** debe establecerse en un estado lógico uno para iniciar la conversión ADC. Este bit permanece en nivel lógico alto mientras la conversión está en curso y el hardware lo borra una vez que se completa la conversión.

La primera conversión después de encender el ADC requiere 25 ciclos de reloj del ADC para inicializar el circuito analógico. Luego, para conversiones posteriores, se necesitan 13 ciclos de reloj ADC (13,5 para conversiones activadas automáticamente).

## Proyecto de muestra

Un proyecto de muestra para usar el sensor de temperatura está disponible aquí. (/8avr:avradc)

# Capacitación sobre ADC y optimización de energía para microcontroladores 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 presentan diferentes técnicas para demostrar cómo se puede reducir el consumo de corriente en los microcontroladores de las series **tinyAVR® 0 y 1** y **megaAVR® 0**.

Este tutorial también demuestra 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 para todos los dispositivos tinyAVR 0- y 1-series, y megaAVR 0-series.

Los proyectos de solución para cada una de las asignaciones están disponibles en el navegador de ejemplo Atmel START (<http://start.atmel.com/#examples>).

En la categoría 'Primeros pasos', busque ADC y solución de optimización de energía (1-5).

Los enlaces directos a los proyectos de ejemplo relevantes se proporcionan en las descripciones de tareas a continuación.

- ADC y optimización de energía: manual tutorial práctico  
(<http://ww1.microchip.com/downloads/en/DeviceDoc/40002008A.pdf>)

## Requisitos previos de hardware

- Kit de evaluación ATtiny817 Xplained Pro
- Cable micro-USB (Tipo-A/Micro-B)
- un potenciómetro
- Tres cables macho a hembra
- conexión a Internet

## Requisitos previos del software

- Estudio Atmel 7.0
- Navegador web. La lista de navegadores compatibles se puede encontrar aquí: START navegadores compatibles (<http://start.atmel.com/static/help/index.html?GUID-51435BA6-0D59-4458-A413-08A066F6F7CA>)

**Tiempo estimado de finalización: 120 minutos**

## Tarea 1: Conversión de ADC con la aplicación de impresión USART (<http://alexandria.atmel.com/keyword/AVR.TRAINING.ADCPOWER.ASSIGN1/redirect>)

En esta tarea, se utiliza Atmel Studio para desarrollar una aplicación utilizando controladores ADC y USART de Atmel START. El ADC está configurado para funcionar en modo de conversión única y se conecta un potenciómetro al pin de entrada del ADC para estudiar la funcionalidad del ADC. Los datos del 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.



Solución de la tarea  
([http://start.atmel.com/#example/Atmel%3AADC\\_and\\_Power\\_Optimization%3A1.0.0%3A%3AApplication%3AADC\\_and\\_Power\\_Optimization\\_S](http://start.atmel.com/#example/Atmel%3AADC_and_Power_Optimization%3A1.0.0%3A%3AApplication%3AADC_and_Power_Optimization_S))

## Tarea 2: RTC interrumpe los disparadores ADC y USART Imprimir (<http://alexandria.atmel.com/keyword/AVR.TRAINING.ADCPOWER.ASSIGN2/redirect>)

En esta asignación, se utiliza el módulo Contador en tiempo real (RTC). La interrupción de desbordamiento de RTC se utiliza para activar una conversión de ADC cada medio segundo. La interrupción ADC Result Ready (RESRDY) activa una impresión del resultado ADC en el terminal USART. Cuando la interrupción de desbordamiento de RTC no se activa, el dispositivo se mantiene en modo de suspensión en 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. Posteriormente, se regenera un proyecto de Atmel Studio.



Solución de la tarea [tarea](http://start.atmel.com/#example/Atmel%3AADC_and_Power_Optimization%3A1.0.0%3A%3AApplication%3AADC_and_Power_Optimization_S)

### Tarea 3: Optimización de energía en pines de E/S

(<http://alexandria.atmel.com/keyword/AVR.TRAINING.ADCPOWER.ASSIGN3/redirect>)

En esta asignación, el búfer de entrada digital en los pines de E/S está deshabilitado 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 el período sin 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.



Solución de la tarea [tarea](http://start.atmel.com/#example/Atmel%3AADC_and_Power_Optimization%3A1.0.0%3A%3AApplication%3AADC_and_Power_Optimization_S)

### Tarea 4: Conversión de ADC usando el modo de comparación de ventana

(<http://alexandria.atmel.com/keyword/AVR.TRAINING.ADCPOWER.ASSIGN4/redirect>)

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



Solución de la tarea [tarea](http://start.atmel.com/#example/Atmel%3AADC_and_Power_Optimization%3A1.0.0%3A%3AApplication%3AADC_and_Power_Optimization_S)

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

(<http://alexandria.atmel.com/keyword/AVR.TRAINING.ADCPOWER.ASSIGN5/redirect>)

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



Solución de la tarea [tarea](http://start.atmel.com/#example/Atmel%3AADC_and_Power_Optimization%3A1.0.0%3A%3AApplication%3AADC_and_Power_Optimization_S)

## 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 sencilla aplicación de conversión ADC y se introducen diferentes técnicas para demostrar cómo se puede reducir el consumo de corriente. Esta es una base útil para desarrollar futuras aplicaciones ADC con requisitos específicos de consumo de corriente.

# Descripción general de la operación de bajo consumo de AVR® MCU

## Descripción general de bajo consumo

El microcontrolador AVR® de 8 bits proporciona varios modos de reposo y sincronización 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 utilizan interrupciones o reinicio para reactivar el dispositivo nuevamente. El reloj individual de los periféricos no utilizados se puede detener durante el funcionamiento normal o en suspensión, lo que permite una gestión de energía mucho más precisa que los modos de suspensión por sí solos.

Para llegar a 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 energía, debe considerar usar el voltaje del sistema más bajo posible. Además, el consumo también es directamente proporcional a la frecuencia del reloj y, si no se utilizan los modos de suspensión, el dispositivo debe funcionar con la frecuencia más baja posible.

### Consejos y trucos para reducir la potencia de un AVR®

- Use el **registro de reducción de energía (PRR0)** para detener el reloj en 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

(/local--files/8avr:low-power-overview/prp-picture.png)

#### 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

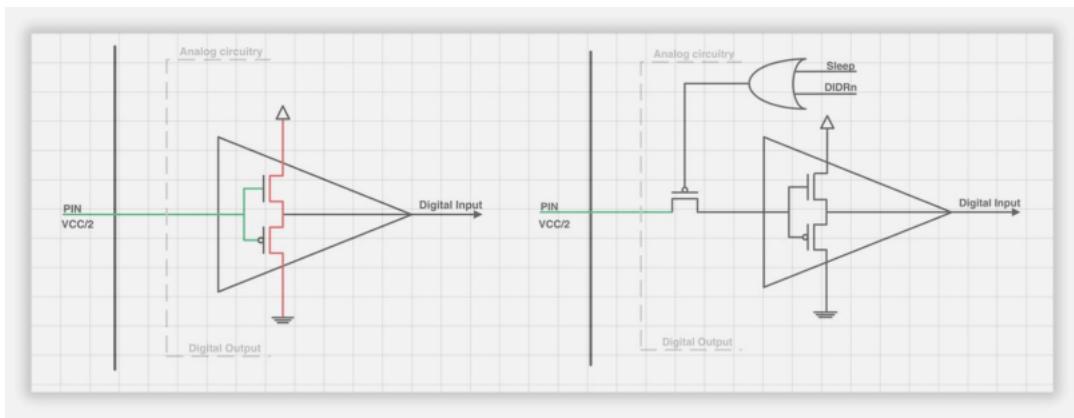
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

#### 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.

(/local--files/8avr:low-power-overview/prt-description.png)

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



(/local--files/8avr:low-power-overview/didr-circuit.png)

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

Bit	7	6	5	4	3	2	1	0
	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
Access	R/W							
Reset	0	0	0	0	0	0	0	0

(/local--files/8avr:low-power-overview/didr0-picture.png)

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

Bit	7	6	5	4	3	2	1	0
							AIN1D	AIN0D
Access							R/W	R/W
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 AIN1/0 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 AIN1/0 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.

(/local--files/8avr:low-power-overview/didr1.png)



Visite la página ([/8avr:low-power-example](#)) de ejemplo de bajo consumo para ver un código de ejemplo sobre cómo reducir el consumo de energía del AVR.

# Proyecto de ejemplo de AVR de baja potencia

## ◎ Objetivo

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

- Cree un proyecto y agregue un código simple para hacer parpadear un LED
  - Ejecutar el proyecto y medir el consumo de energía
  - Hacer modificaciones al código para reducir la potencia
  - Ejecute el proyecto modificado y observe un consumo de energía reducido

## **Materiales**

## Requisitos de Software

Instaladores

---

Windows  Linux  Mac OS X  Instrucciones de instalación 

---

Herramienta  Sobre

---

Entorno de desarrollo integrado Atmel® Studio  (/atstudio:start)  (<http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe>)    (/install:atstudio)

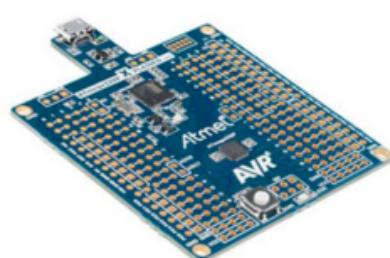
## Requisitos de hardware

- ATmega328PB Mini tablero explicado
  - Kit depurador de energía
  - Dos cables micro USB
  - Tres cables hembra a macho y un cable macho a macho

Herramienta	Sobre
 ( <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">http://www.atmel.com/tools/MEGA328PB-XMINI.aspx</a> )	Mini kit de evaluación <b>ATmega328PB-Xplained</b>  ( <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">http://www.atmel.com/tools/MEGA328PB-XMINI.aspx</a> )  ( <a href="https://www.microchipdirect.com">https://www.microchipdirect.com</a> )

ATmega328PB Tarjeta explicada

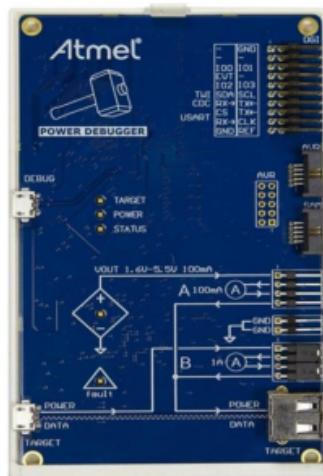
El **mini kit de evaluación ATmega328PB Xplained** es una plataforma de hardware para evaluar el microcontrolador Atmel ATmega328PB. NO se necesita un depurador externo para ejecutar estos ejercicios. El ATmega328PB tiene un depurador incorporado completamente integrado.



(/local--files/8avr:low-power-example/xplained-board.png)

## Kit depurador de energía

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 depuración de aplicaciones y medidas de energía en tiempo de ejecución al visualizador de datos.



(/local--files/8avr:low-power-example/power-debug.png)

El Power Debugger 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 corrientes bajas.

El canal 'B' es el canal recomendado para medir corrientes más altas con una resolución más baja.

## configuración de hardware

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

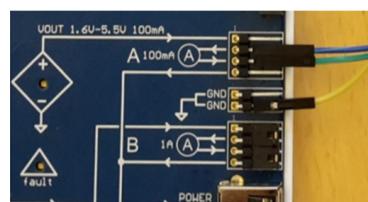
Los puertos de medición de corriente de los canales 'A' y 'B' en la placa Power Debugger se representan con símbolos de amperímetro en la serigrafía. El suministro de voltaje está conectado a la entrada del amperímetro y la carga (objetivo) está conectada a la salida. Con los siguientes pasos, el depurador de energía mide el consumo en el núcleo AVR.

Tabla 1-1 Power Debugger y ATmega328PB Explicación de la conexión Mini.

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

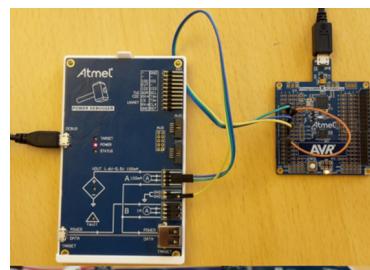
(/local--files/8avr:low-power-example/table.png)

Figura 1-1 Conexión de Power Debugger y ATmega328PB Xplained Mini .



(/local--files/8avr:low-power-example/connection.png)

### Configuración de hardware



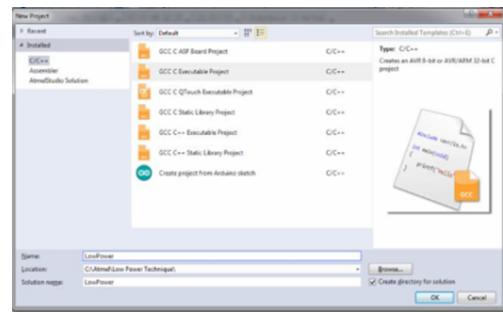
(/local--files/8avr:low-power-example/hw.png)

## Medición de la corriente en modo activo

### ! Procedimiento

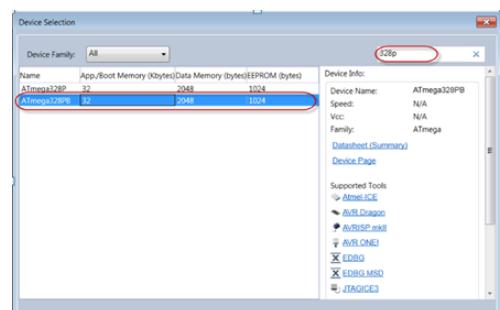
#### A Creación de proyectos

- Abrir Atmel Studio 7
- Seleccione **Archivo > Nuevo > Proyecto**
- Seleccione **GCC C Executable Project** y asignele el nombre **LowPower** .
- Elija una ubicación para guardar el proyecto en su computadora.



(/local--files/8avr:low-power-example/location.png)

- Cuando aparezca la ventana de selección de dispositivo, ingrese 328P en la ventana de búsqueda y luego seleccione Atmega328PB. Haga clic en Aceptar.



(/local--files/8avr:low-power-example/device-selection.png)

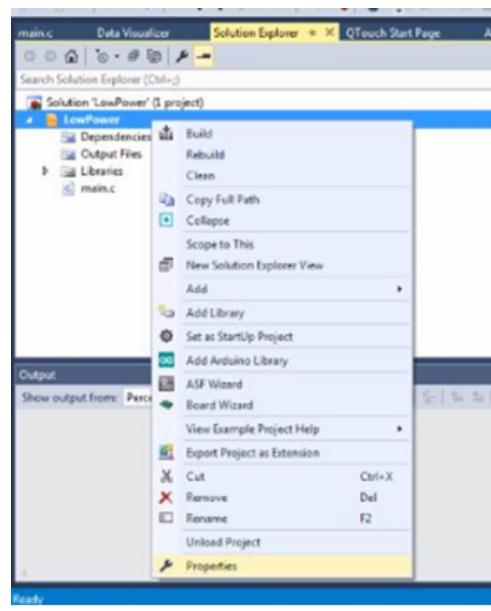
- Se creará un proyecto que contiene un `ciclo while(1)` vacío en `main()` .

```

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

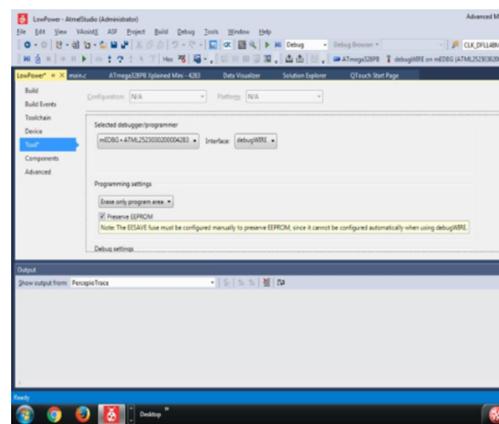
Seleccione **Ver > Explorador** de soluciones en la barra de menú.

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



(/local--files/8avr:low-power-example/se.png)

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



(/local--files/8avr:low-power-example/debugwire.png)

## B Agregar código al proyecto

- Agregue las siguientes dos declaraciones en `main()` para llamar a `TMR0_init()` y establecer un pin de E/S como salida:

```
File: main.c
TMR0_init();
DDRB |= 1<<DDRB5; // Direction o
```

Agregue la rutina de inicialización del temporizador 0

```
File: main.c
TMR0_init();
void TMR0_init( void ){
    // enable timer overflow inter
    TIMSK0=(1<<TOIE0)
    // set timer0 counter initial
    TCNT0=0x00;
    // start timer0 with /1024 pre
    TCCR0B = (1<<CS02) | (1<<CS00)

    // enable interrupts
    sei();
}
```

- Proporcione la rutina de servicio de interrupción TMR0 al proyecto y haga que el LED parpadee a aproximadamente 1 Hz.

```
uint8_t count;
ISR(TIMER0_OVF_vect){
    count++;
    if(count==20){
        count=0;
        // Toggle 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(TIMER0_OVF_vect){
    count++;
    if(count==20){
        count=0;
        // Toggle LED
        PORTB=PORTB ^ 0x20;
    }
}

int main(void)
{
    TMR0_init();

    DDRB |= 1<<DDRB5; // Direct
    /* Replace with your applicati
    while (1)
    {
    }

/*****
    TM
*****
void TMR0_init( void ){
    // enable timer overflow inter
    TIMSK0=(1 <<TOIE0) ;
    // set timer0 counter initial
    TCNT0=0x00;
    // start timer0 with /1024 pre
    TCCR0B = (1 <<CS02) | (1<<CS00
    // enable interrupts
    sei();
}
```

## C Verificar las ejecuciones del proyecto

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



El LED parpadeará si el proyecto funciona correctamente

- Asegúrese de que la depuración del proyecto finalice seleccionando **Depurar > Deshabilitar debugWire** y luego **Cerrar**.

## D Medir el consumo de energía en modo activo

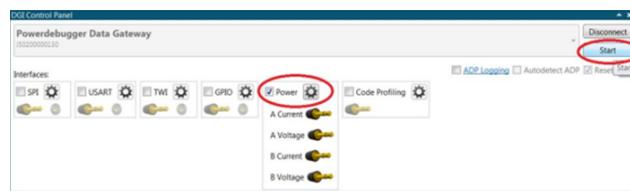
- En Atmel Studio 7, abra el menú **Herramientas > Visualizador de datos**

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

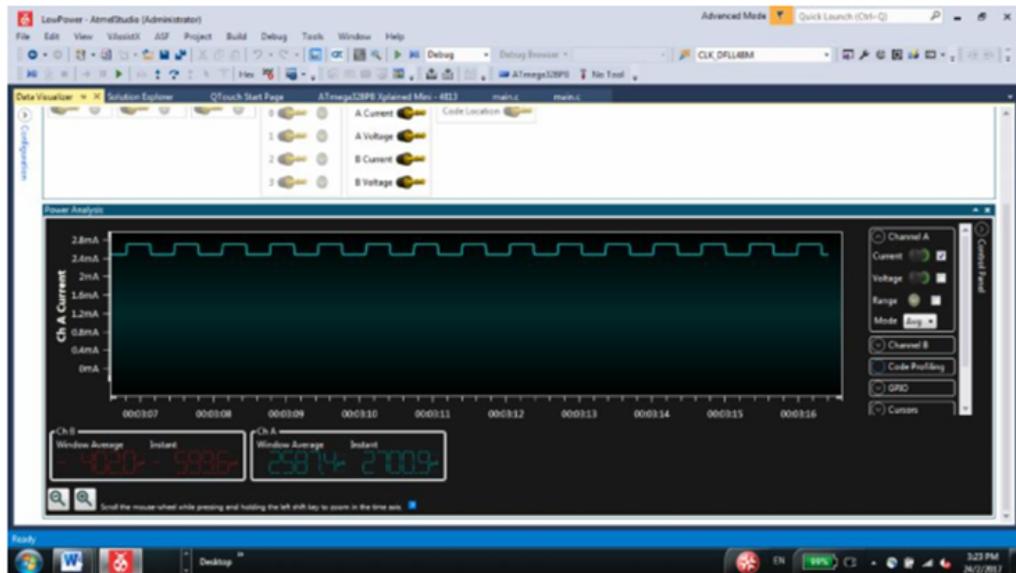


(/local--files/8avr:low-power-example/connect.png)

- Seleccione **Encendido y arranque**

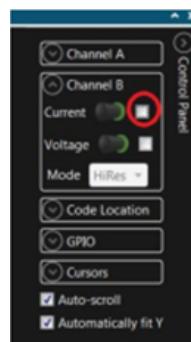


(/local--files/8avr:low-power-example/power-and-start.png)



(/local--files/8avr:low-power-example/power-monitor.png)

Apague la fuente de alimentación de destino a la placa explicada y habilite la medición de energía



(/local--  
files/8avr:low-  
power-  
example/enable-  
power.png)

- Cierre la programación del dispositivo
- Verifique que el consumo medio de corriente sea de unos 2,6 mA

## Reducir el consumo de energía

Existen varios métodos que puede utilizar para reducir el consumo de energía de un microcontrolador AVR®. Este ejemplo reduce el poder por:

- apagar los periféricos no utilizados
- Detener la corriente de fuga en los pines de E/S digital
- 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 usa como pin analógico, no hay necesidad de saber si el valor digital de la señal analógica sería uno o cero; estos pines digitales deben estar deshabilitados.

## Apague los periféricos no utilizados

Deshabilite los periféricos no utilizados en la aplicación. El **registro de reducción de energía ( PRR0 )** y ( **PRR1** ) pueden detener el reloj de periféricos individuales para reducir el consumo de energía. Los recursos utilizados por el periférico permanecen ocupados cuando se detiene el reloj. 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 código a continuación para `optimizar_el_consumo_de_energía()`.



```
/* Power shutdown to unused peripherals */
PPR0 = 0xDF;
PPR1 = 0x3F;
```

3. Agregue el `#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 desconectados consumen energía. La carga de energía innecesaria de los pines flotantes se puede evitar usando las resistencias pull-up internas del AVR en los pines no utilizados. Los pines del puerto se pueden configurar para ingresar pull-ups configurando el bit **DDxn** en 0 y el bit **PORTxn** en 1. (donde x es PORT B, C, D, E y n es 0 a 7)

```
1  /* Unused pins set as inputs */
2  DDRB  &= 0xE0;
3  DDRC &= 0xC9;
4  DDRD &= 0x03;
5  DDRE &= 0xF3;
6
7  PORTB |= ~(0xE0);
8  PORTC |= ~(0xC9);
9  PORTD |= ~(0x03);
10  PORTE |= ~(0xF3);
```

## Usar la función de suspensión

El modo de suspensión permite que la aplicación ahorre energía al apagar 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 el archivo de encabezado <avr/sleep.h>** de la biblioteca AVR en la parte superior del archivo.
2. Establezca el modo de suspensión deseado en la función `Optimize_Power_consumption()` configurando el microcontrolador para usar el modo de suspensión **SLEEP\_MODE\_PWR\_DOWN** para un consumo mínimo de energía.

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

3. Llame a la función `sleep_mode()` desde `main()` para ingresar al modo de suspensión.



```

1  while (1)      ?
2  {
3      sleep_mode();
4 }
```

## Uso de la interrupción de cambio de pin

Para despertar el microcontrolador de **SLEEP\_MODE\_PWR\_DOWN**, se utiliza la **interrupción de cambio de pin**. El interruptor de la **mini placa ATmega328PB Xplained** (SW0) está conectado a **PB7**. El pin **PB7** es la fuente de la interrupción por cambio de pin. Haz las siguientes adiciones a `main()`:

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

Configuración del bit PCINT7 y registro PCICR:

```

PCMSK0 |= (1<<PCINT7); //Enable f2
PCICR |= (1<<PCIE0); //Enable t
sei();
```

## Agregue la rutina de servicio de interrupción:

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

```

ISR (PCINT0_vect)
{
    if (!(PINB & (1<<PINB7))) //
    {
        PORTB |= (1 <<PORTB5); //
    }
    else
    {
        PORTB &= ~(1<<PORTB5); //
    }
}
```

## Código completado

Después de realizar los cambios anteriores, el código completo debería ser como el siguiente:

```

/*
 * lowpower2.c
 */
#include <avr/io.h>

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

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

void optimize_power_consumption

/* Disable digital input buffer
DIDR0 = (1 << ADC5D) | (1 << A
DIDR0 |= 0xC0 ; /*ADC7D and

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

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

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 */
WDTCR = 0x00;

/* Set sleep mode */
set_sleep_mode(SLEEP_MODE_PWR_

}

***** TM *****
void TMR0_init( void ){

    // enable timer overflow inter
    TIMSK0=(1<<TOIE0) ;

    // set timer0 counter initial
    TCNT0=0x00;

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

    // enable interrupts
    sei();

}

    uint8_t    count;
/* Timer 0 Interrupt */
ISR(TIMER0_OVF_vect){

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

ISR (PCINT0_vect)
{
    if (!(PINB & (1<<PINB7))) //
    {
        PORTB |= (1<<PORTB5); // T
    }
    else
    {
        PORTB &= ~(1<<PORTB5); //
    }
}

int main(void)
{
    TMR0_init();

    DDRB |= 1<<DDRB5; // Directi
    DDRB &= ~(1<<DDB7); //Set PORT

    optimize_power_consumption();

    PCMSK0 |= (1<<PCINT7); //Enabl
    PCICR |= (1<<PCIE0); //Enable
    sei();

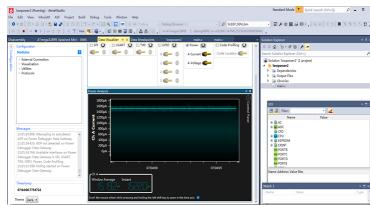
    //set_sleep_mode(SLEEP_MODE_PW
    //sleep_enable();
    //sleep_cpu();

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

```

## Programe la aplicación y mida el consumo de energía.

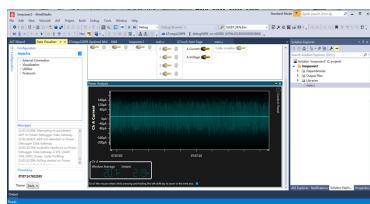
1. Programe el código seleccionando **Depurar > Continuar** .
2. Espere hasta que la aplicación esté programada y el mensaje en la parte inferior de la ventana aparezca como **En ejecución** .
3. Salga del depurador seleccionando **Depurar > Deshabilitar debugWire** y luego **Cerrar** .
4. Abra la **ventana**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



(/local--files/8avr:low-power-example/one-five.png)

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

- Establezca el interruptor de alimentación objetivo apagado: **Herramientas > Programación de dispositivos**  
> Configuración de herramientas



(/local--files/8avr:low-power-example/threema.png)

El consumo de corriente debería haberse reducido a aproximadamente 20,7  $\mu$ A.



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

## 💡 Conclusiones

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

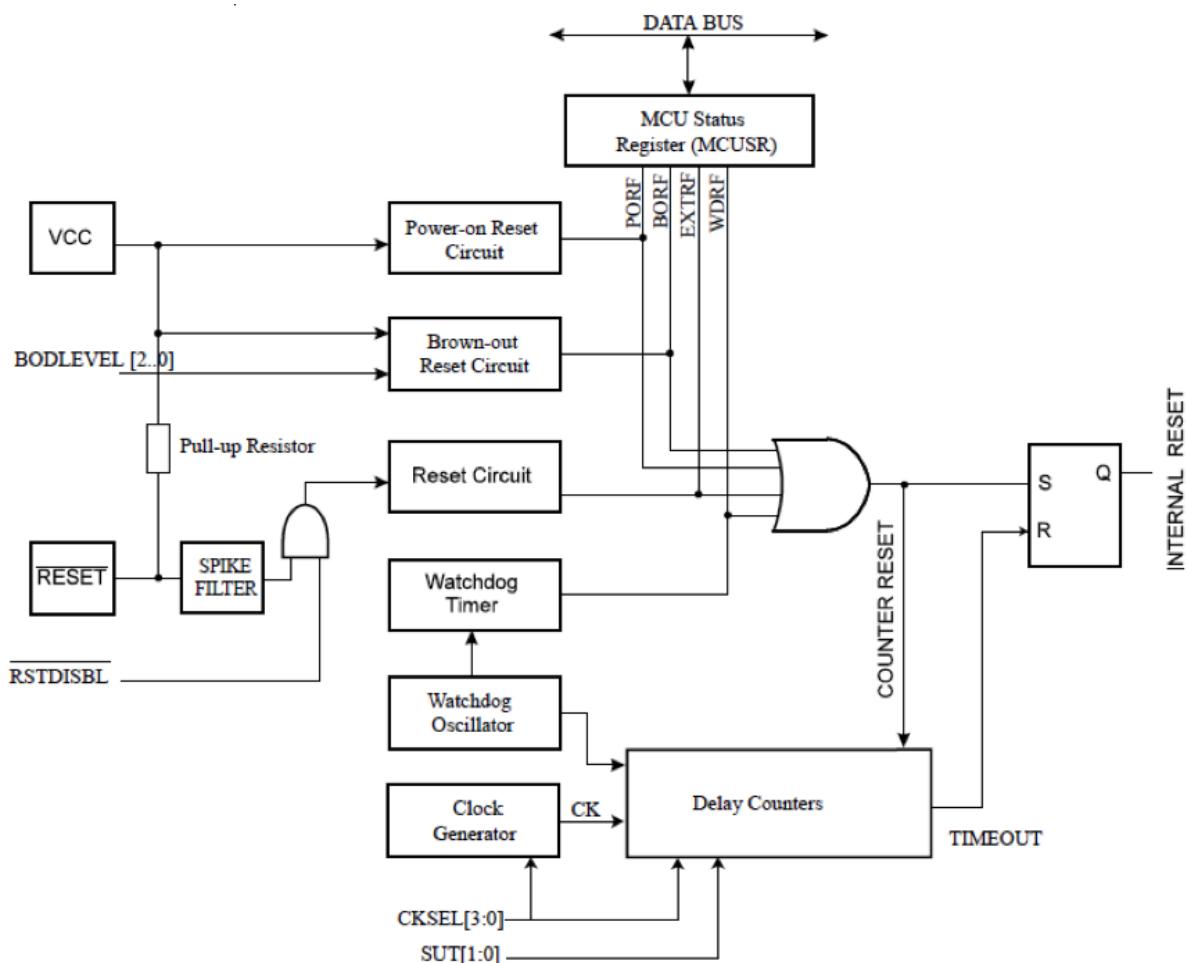
- Diseño inteligente
- Uso de los modos de suspensión
- Desactivación de periféricos no utilizados

Este ejemplo usó el **visualizador de datos Atmel Studio 7** y la placa de **depuración de energía** para medir la energía.

# Fuentes de restablecimiento de AVR

El dispositivo AVR tiene cuatro fuentes de reinicio:

- **Reinicio de encendido** : el microcontrolador (MCU) se reinicia cuando el voltaje de suministro es inferior al umbral de reinicio de encendido (VPOT).
- **Restablecimiento externo** : la MCU se restablece cuando hay un nivel bajo en el pin RESET durante más tiempo que la duración mínima del pulso.
- **Restablecimiento del sistema de vigilancia** : la MCU se restablece cuando expira el período del temporizador de vigilancia y se habilita el modo de restablecimiento del sistema de vigilancia.
- **Restablecimiento por caída de tensión**: la MCU se restablece cuando el voltaje de suministro  $V_{CC}$  es menor que el restablecimiento por caída de tensión.



(/local--files/8avr:avrreset/avr\_reset.png)

## Registro de estado de MCU (MCUSR)

Para hacer uso de los indicadores de reinicio para identificar una condición de reinicio, el usuario debe leer y luego reiniciar el MCUSR lo antes posible en el programa. Si el registro se borra antes de que ocurra otro restablecimiento, la fuente del restablecimiento se puede encontrar 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
Access					WDRF	BORF	EXTRF	PORF
Reset					R/W	R/W	R/W	R/W

### 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.

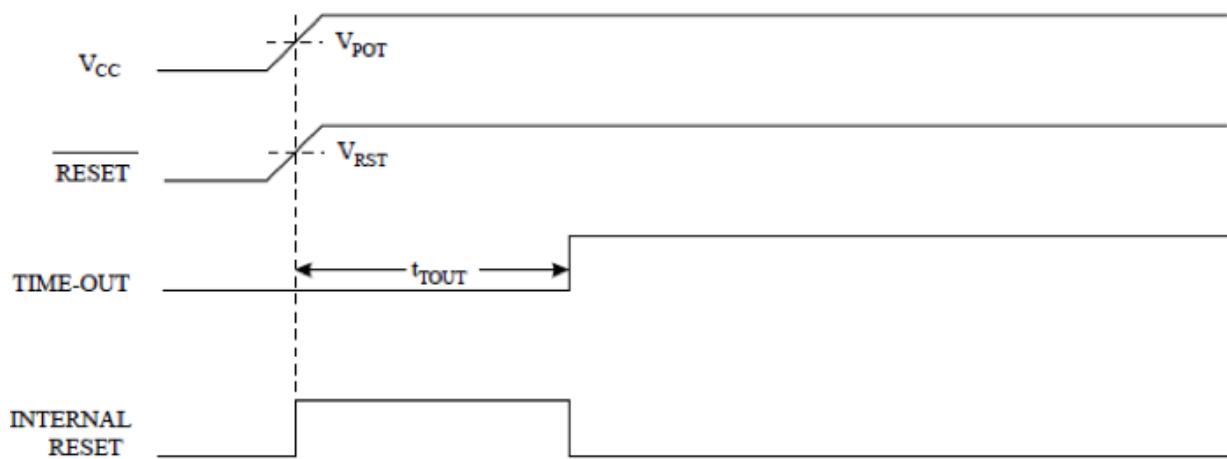
(/local--files/8avr:avrreset/mcusr.png)

## 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 suministro.

Un circuito POR asegura que el dispositivo se restablece desde el encendido. Alcanzar el voltaje de umbral de reinicio de encendido invoca el contador de retardo, que determina cuánto tiempo se mantiene el dispositivo en reinicio después del aumento de  $V_{CC}$ .

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



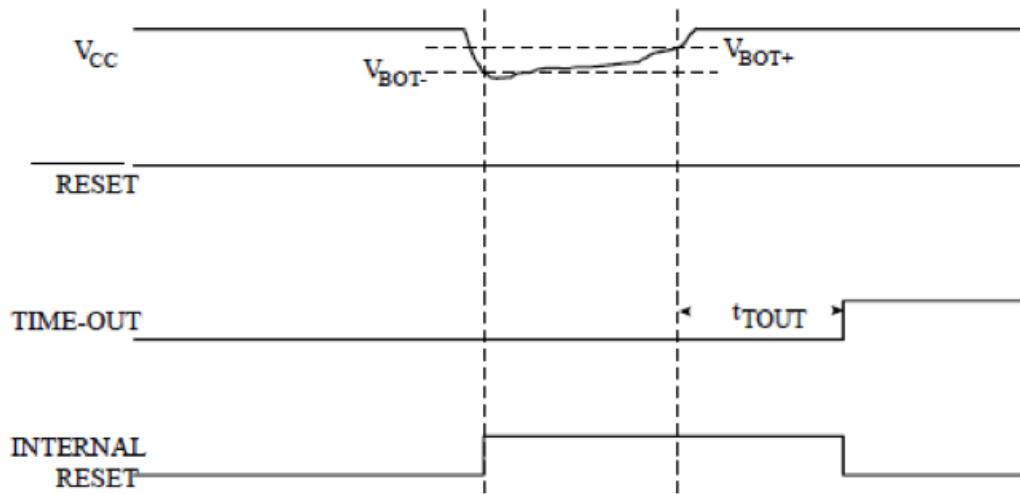
(/local--files/8avr:avrreset/powerup.png)

## Detección de caída de tensión (BOD) y Restablecimiento de caída de tensión (BOR)

El dispositivo tiene un circuito BOD en chip para monitorear el nivel de  $V_{CC}$  durante la operación comparándolo con un nivel de disparo fijo. El nivel de activación para el BOD se puede seleccionar mediante 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 ( $V_{BOT-}$ ) por 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 la MCU después de que haya expirado el período de tiempo de espera  $t_{TOUT}$ .



(/local--files/8avr:avrreset/bor.png)

## Temporizador de vigilancia (WDT)

El WDT se ejecuta independientemente del resto del sistema, lo que hace que el sistema se reinicie cada vez que se agote el tiempo de espera. Sin embargo, el software de la aplicación debe garantizar que nunca se agote el tiempo de espera reiniciando el WDT periódicamente siempre que el software se encuentre en un estado saludable conocido. Si el sistema se bloquea o la ejecución del programa se corrompe, el WDT no recibirá su reinicio periódico y, finalmente, expirará y provocará un reinicio del sistema.

El WDT mejorado en algunos dispositivos AVR también tiene la capacidad de generar interrupciones en lugar de reiniciar el dispositivo. Dado que el WDT funciona con su propio reloj independiente, se puede utilizar para activar el AVR desde todos los modos de suspensión. Esto lo convierte en un temporizador de despertador ideal, que se combina fácilmente con el funcionamiento normal como fuente de reinicio 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 una memoria no volátil.

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

El CFD permite al usuario monitorear el oscilador de cristal de baja potencia o la señal del reloj externo (XOSC). El XOSC es monitoreado por el circuito CFD que opera con el oscilador interno de 128kHz. CFD supervisa 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 del sistema RC interno de 8 MHz. Esto permite configurar el reloj seguro para satisfacer las necesidades de seguridad de la aplicación.

# Proyecto de ejemplo de fuentes de restablecimiento de AVR®

## ⌚ Objetivo

Este proyecto pasa por varias condiciones de reinicio diferentes: reinicio de encendido (POR) , reinicio de apagado (BOR) y tiempo de espera del temporizador de vigilancia (WDT) , y muestra cómo funciona cada uno en una placa Xplained de 328 PB . 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 las fuentes de restablecimiento de AVR. (/avr:avrreset)

## ☒ Materiales

### Herramientas de hardware ( *opcional* )

Herramienta	Sobre
 (http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	Mini kit de evaluación ATmega328PB Xplained <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">Sobre</a> (https://www.microchipdirect.co

### Herramientas de software

Herramienta	Sobre	Instaladores			Instrucciones de instalación
		Windows	Linux	Mac OS X	
 Entorno de desarrollo integrado Atmel® Studio	<a href="#">Sobre</a> (atstudio:start) <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">Descargar</a>	<a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">Descargar</a>	<a href="#">Descargar</a>	<a href="#">Descargar</a>	<a href="#">(/install:atstudi</a>

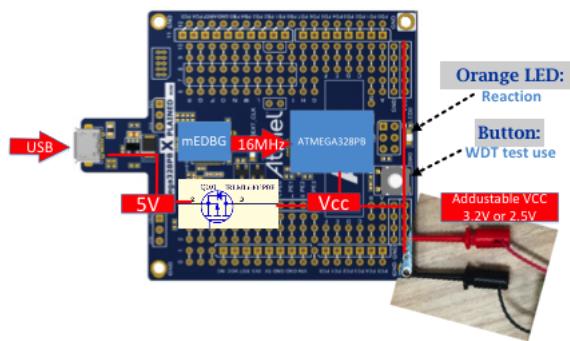
### Archivos de ejercicios

Expediente	Windows
 Archivos de proyecto y fuente	<a href="https://microchiptechnology.sharepoint.com/:u/s/DeveloperHelp/ER6vOhSPHdIGtX1waUOOSH4BJ-IfY1jL6aKVwgc2Q8MizA?e=3PW9UP">Descargar</a>

### Archivos adicionales

archivos
 <a href="http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42469-ATmega328PB-Xplained-Mini_User-Guide.pdf">Guía del usuario de la miniplaca Xplained 328PB</a> (http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42469-ATmega328PB-Xplained-Mini_User-Guide.pdf)

## 🔧 Diagrama de conexión

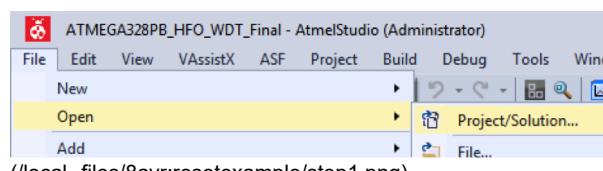


(/local--files/8avr:resetexample/project1.png)

## Procedimiento

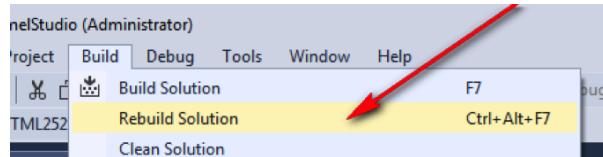
### 1 Tarea 1: abrir y compilar el proyecto

- Descargue los archivos fuente del proyecto de la sección anterior Archivos de ejercicios y descomprimalos en su computadora.
- Abrir Atmel Studio 7
- Seleccione Archivo > Abrir > Proyecto/Solución



(/local--files/8avr:resetexample/step1.png)

- Seleccione ATMEGA328PB\_HFO\_WDT\_Final.atsln de los archivos de proyecto descargados.
- En el menú Generar, seleccione 'Reconstruir solución'.

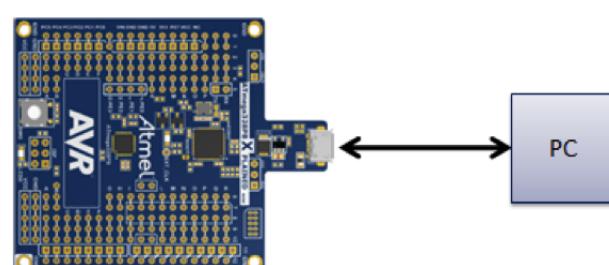


(/local--files/8avr:resetexample/step2.png)

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

### 2 Tarea 2: preparación de la placa

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



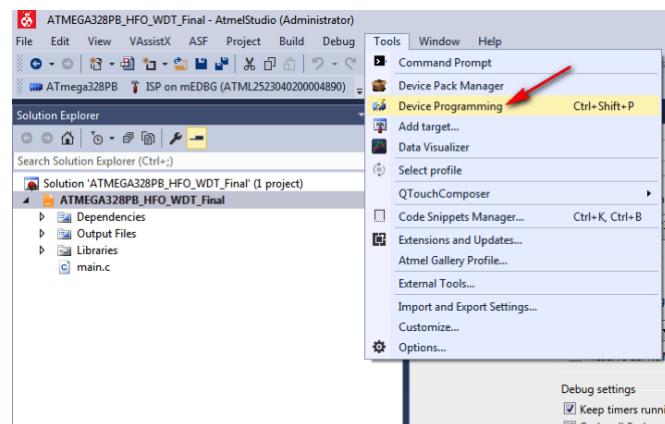
(/local--files/8avr:resetexample/step3.png)

### 3 Tarea 3 - Configuración del programador

Si debugWire está habilitado, desactívelo.

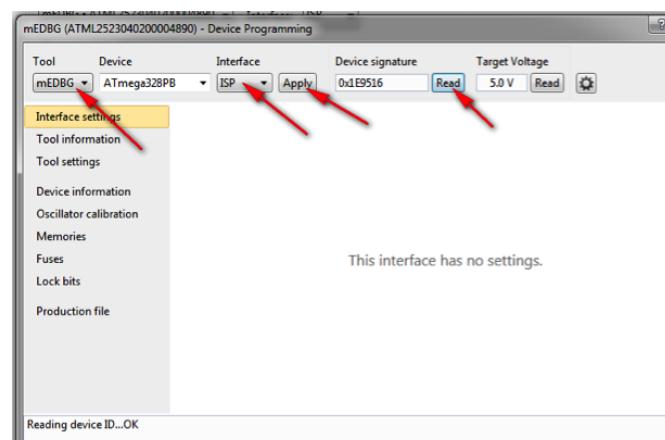
Deshabilitar debugWire (DWEN) ►

- Haga clic en Herramientas > Programación de dispositivos :



(/local--files/8avr:resetexample/step4.png)

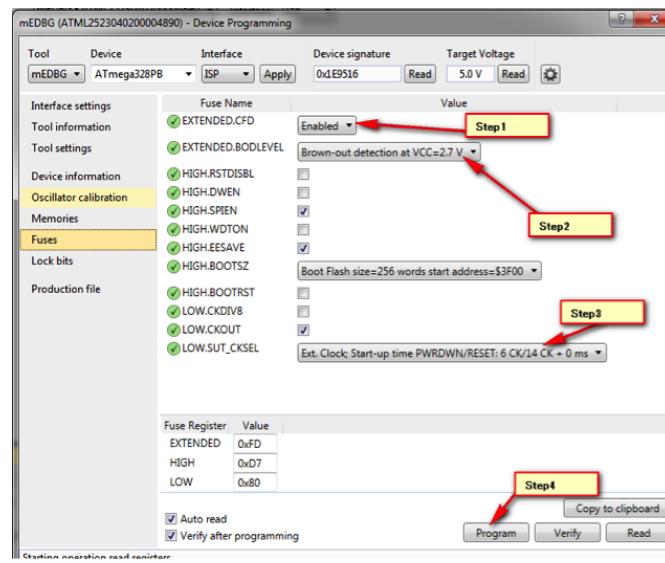
- Haga clic en los botones que están marcados como flechas rojas en la captura de pantalla.



(/local--files/8avr:resetexample/step5.png)

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

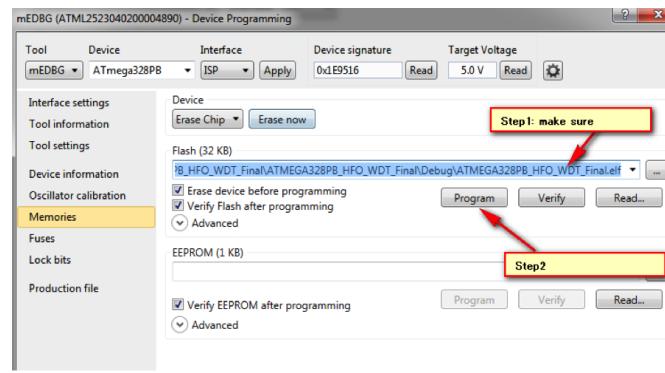
- Seleccione las opciones de 'Fusibles' en la barra de opciones de la izquierda y establezca los bits como se muestra en la figura, lo que habilitará CFD, umbral de DBO como 2,7 V y reloj externo.



(/local--files/8avr:resetexample/step6.png)

#### 5 Tarea 5 - Programación de dispositivos

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



(/local--files/8avr:resetexample/step7.png)

## 6 Tarea 6 - Código del proyecto

Aquí está el código completo al que hacemos referencia.

También puedes descargarlo en la Sección de 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 Re
    set Mode if unintentionally enabled.
    (0<<WDP3)|(1<<WDP2)|(1<<WDP1)|(1<<WDP0);                      // Set Watchdog Timeout peri
    od 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 B0RFbit 2
#define PORFbit 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 Low
            _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 Low
        _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){

    PORTB |= (1 << PINB5);           // Set PORTB5 high
    _delay_ms(250);
    _delay_ms(250);
    _delay_ms(250);
    _delay_ms(250);

    PORTB &= ~(1 << PINB5);          // Set PORTB5 Low
    _delay_ms(250);
    _delay_ms(250);
    _delay_ms(250);
    _delay_ms(250);

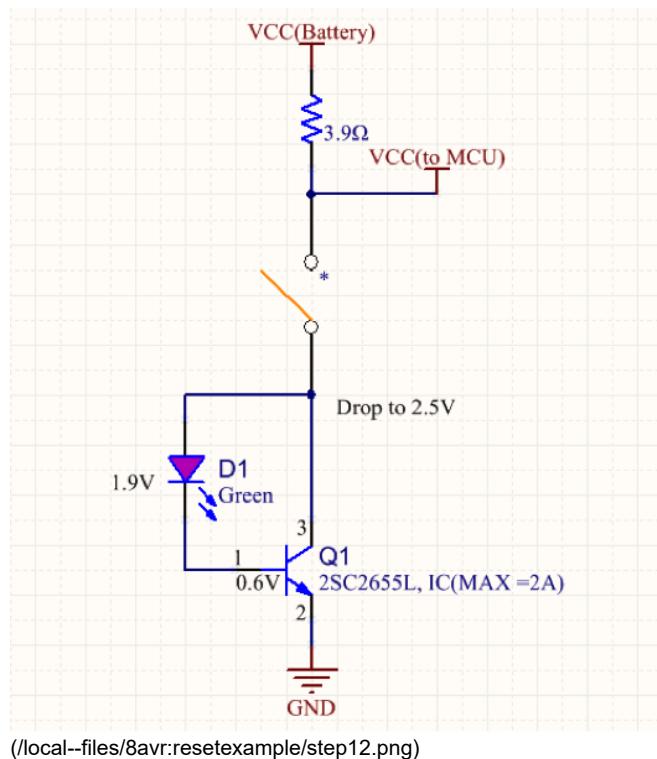
    if((PINB&1<<PINB7)==0){
        PORTB |= (1 << PINB5);           // Set PORTB5 high
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
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

}

```

## 7 Tarea 7: reinicio del circuito de prueba

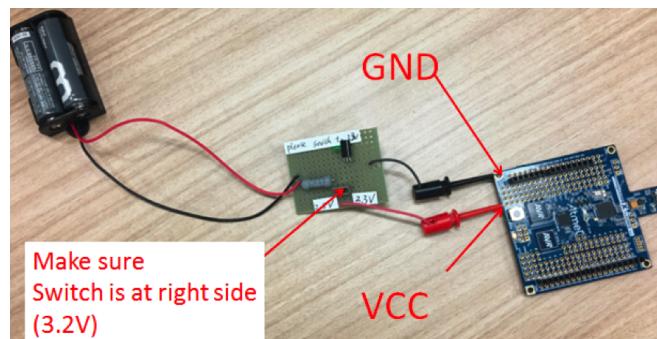
- Construya el circuito de prueba de reinicio que se muestra en el esquema.



(/local--files/8avr:resetexample/step12.png)

## 7 Tarea 8 - Prueba POR

- Saque el cable USB
- Asegúrese de que el interruptor del cable VCC ajustable externo esté en el lado derecho (3,2 V)
- Sujete el cable VCC ajustable externo a la placa



(/local--files/8avr:resetexample/step9.png)

- Resultado: el LED naranja responderá parpadeando (0,3 ms, cuatro veces) según la sección de código a continuación porque se detectó 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;
    }
}
```

(/local--files/8avr:resetexample/step10.png)

## 9 Tarea 9 - Prueba BOR

- Después de la prueba POR, deslice el interruptor del cable VCC ajustable externo hacia el lado izquierdo (2,5 V)
- Deslice el interruptor del cable VCC ajustable externo hacia el lado derecho (3,2 V)
- Resultado: el LED naranja responderá parpadeando (0,1 ms, ocho veces) según el código resaltado a continuación porque se detectó 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);
    }
}
(/local--files/8avr:resetexample/step13.png)

```

## 10 Tarea 10 - Prueba WDT

- Presione el botón en el tablero durante aproximadamente 1 segundo y luego suelte el botón. Se activarán retrasos prolongados en la rutina principal, lo que provocará un tiempo de espera de 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);
}
wdt_reset();
}
(/local--files/8avr:resetexample/step14.png)

```

- Resultado: el LED naranja responderá con un parpadeo rápido (0,02 ms encendido, 0,08 ms apagado, cuatro veces) al principio cuando el WDT interrumpe el 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);
    }
}
(/local--files/8avr:resetexample/step15.png)

```

- Resultado: el LED naranja volverá a responder con un parpadeo rápido (0,02 ms encendido, 0,08 ms apagado, cuatro veces) a medida que se detecta el restablecimiento del 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;
}
(/local--files/8avr:resetexample/step16.png)

```

## Q 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 de cómo se pueden configurar e implementar estos tipos de restablecimientos.

## Conclusiones

El proyecto ayuda a explicar cómo funciona el circuito de reinicio dentro del AVR y cómo implementarlo. La sección de código se puede reutilizar en aplicaciones futuras que pueden requerir una estructura de reinicio similar. De ninguna manera es esta la única forma de diseñar restablecimientos 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.