TECNICATURA SUPERIOR EN TELECOMUNICACIONES

ELECTRÓNICA MICROCONTROLADA

Docentes: Ing. Jorge E. Morales, Téc. Gonzalo Vera.

**Título:** Periféricos AVR de 8 bits.

**Grupo 1:**

❖ Birge, Adolfo Federico.

❖ Carunchio, Carlos Javier.

❖ Ferreyra, María Luciana.

❖ Gutiérrez, Emma Vilma.

❖ Merlo, Emmanuel.

❖ Romero, Gisela de Lourdes.

**Periféricos AVR® de 8 bits**

**Oscilador**

**Descripción general del oscilador megaAVR®**

Los microcontroladores microchip megaAVR® de 8 bits tienen varias opciones de fuente de reloj, seleccionables mediante la programación de los bits de [fusibles](https://microchipdeveloper.com/8avr:avrfuses) CKSEL Flash. Esta discusión es específica para el 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.

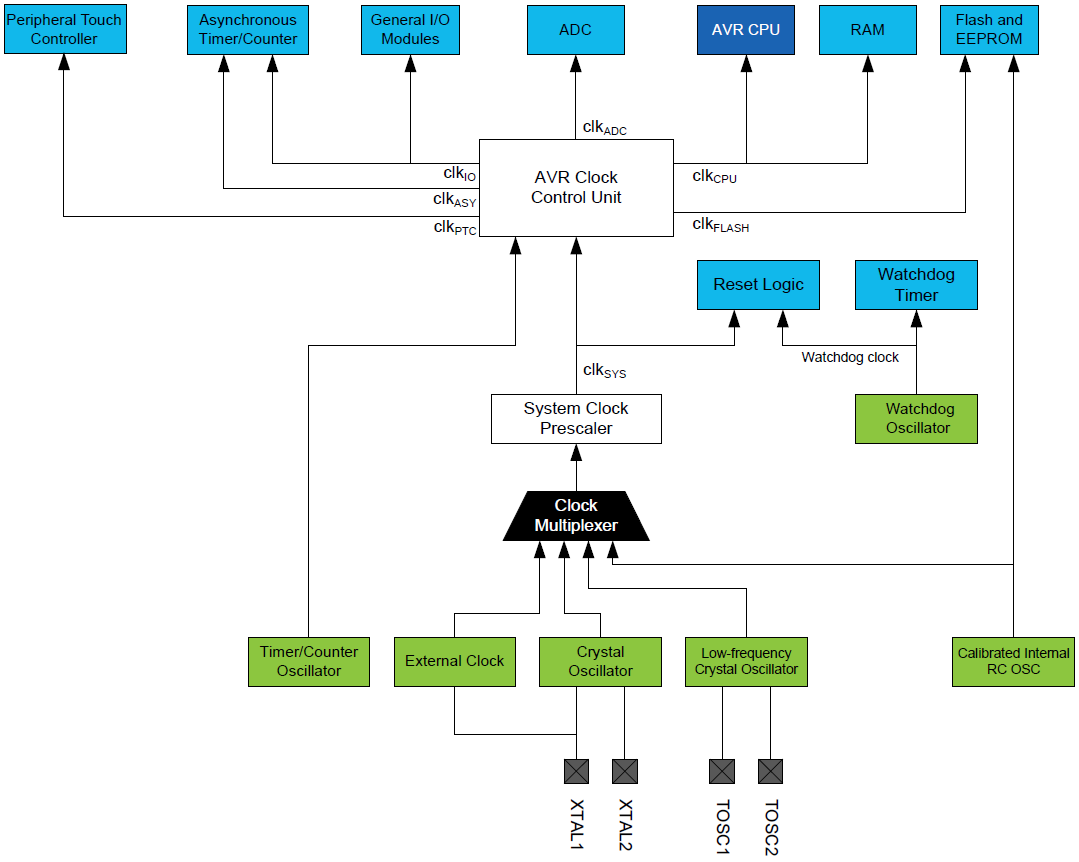
La fuente del reloj del sistema no se puede cambiar durante el tiempo de ejecución, ya que se configura a través de la programación de fusibles.

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

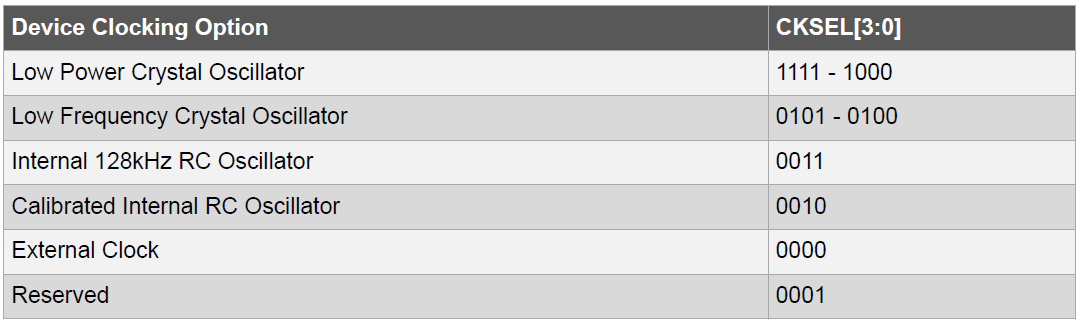
Cada fuente de reloj proporciona una opción de retraso después del restablecimiento o encendido del dispositivo para mantener el dispositivo restablecido hasta que se suministre con un Vcc mínimo. El reloj de la fuente seleccionada se introduce en el generador de reloj AVR® y se enruta a los módulos apropiados.  
La frecuencia máxima de funcionamiento de megaAVR® depende de VCc. El software de aplicación debe asegurarse de que la frecuencia de la fuente de reloj seleccionada se encuentra dentro del área de operación segura (ver 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 en el dispositivo y su distribución. No es necesario que todos los relojes estén activos a una hora determinada. Con el fin de reducir el consumo de energía, los relojes de los módulos que no se utilizan se pueden detener mediante [el uso de diferentes modos de suspensión](https://microchipdeveloper.com/8avr:avrsleep). Los sistemas de reloj se describen en las siguientes secciones. La frecuencia de reloj del sistema se refiere a la frecuencia generada a partir del preescalador de reloj del sistema. Todas las salidas de reloj de la unidad de control de reloj AVR funcionan a la misma frecuencia.

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-sys-clk-distribution.png)

**Fuentes de reloj**

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-clk-sources.png)El dispositivo tiene las siguientes opciones de fuente de reloj, seleccionables a través de bits **CKSEL** Flash Fuse como se muestra a continuación. El reloj de la fuente seleccionada se introduce en el generador de reloj AVR® y se enruta a los módulos apropiados.

**Origen de reloj predeterminado**

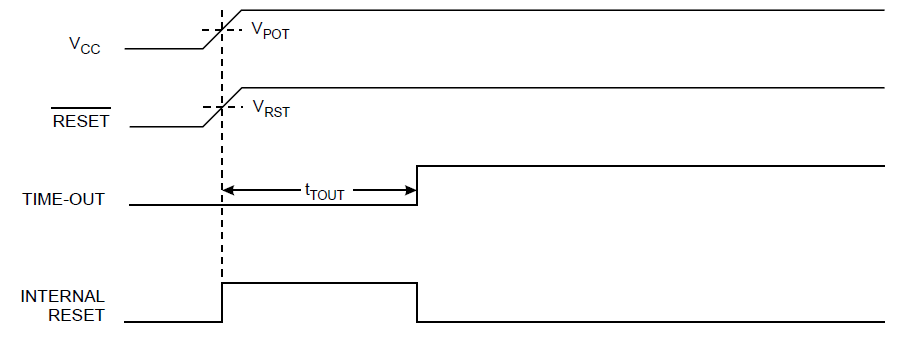
El dispositivo se envía con el oscilador RC interno seleccionado a 8.0 MHz y con el fusible CKDIV8 programado, lo que resulta en un reloj del sistema de 1.0 MHz. El tiempo de inicio se establece en máximo y el período de tiempo de espera está habilitado: CKSEL=0010, SUT=10, CKDIV8=0. Esta configuración predeterminada garantiza que todos los usuarios puedan realizar la configuración de origen de reloj deseada utilizando cualquier interfaz de programación disponible.

**Secuencia de inicio del reloj**

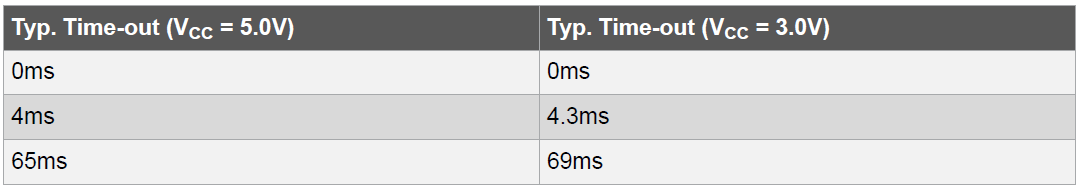
Cualquier fuente de reloj necesita (i) una V suficienteCc para empezar a oscilar y (ii) un número mínimo de ciclos oscilantes antes de que pueda considerarse estable.

**Estabilidad de Vcc**

Para garantizar una VCc, el dispositivo emite un restablecimiento interno con un retraso de tiempo de espera (**tTOUT**) después de que el restablecimiento del dispositivo sea liberado por todas las demás fuentes de restablecimiento:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-tout-delay.png)

El retraso (**tOUT**) se cronometra desde el oscilador Watchdog y el tiempo de retardo se establece mediante los bits de fusible SUTx y CKSELx. Los retrasos seleccionables para t**OUT** se muestran en la siguiente tabla. Tenga en cuenta que la frecuencia del oscilador Watchdog depende del voltaje:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-tout-values.png)

​VCc no se supervisa durante el retraso, por lo que es necesario seleccionar un retraso superior al VCc tiempo de ascenso. Si esto no es posible, se debe utilizar un circuito interno o externo de detección de apagado (DBO). Un circuito BOD asegurará suficiente VCc antes de que se libere el restablecimiento, y el retraso de tiempo de espera se puede deshabilitar. No se recomienda deshabilitar el retardo de tiempo de espera sin utilizar un circuito de detección de salida marrón.

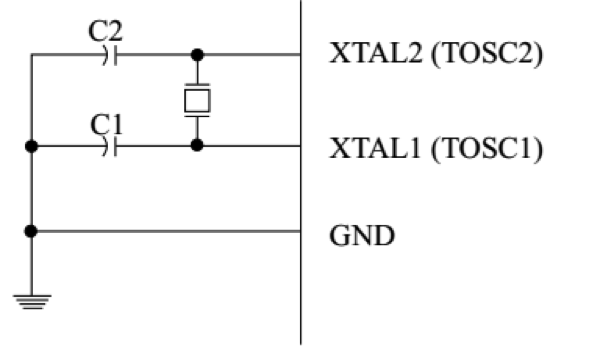
**Estabilidad del oscilador**

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. El restablecimiento se libera y el dispositivo comenzará a ejecutarse. El tiempo de arranque del oscilador recomendado depende del tipo de reloj y varía de 6 ciclos para un reloj aplicado externamente a 32K ciclos para un cristal de baja frecuencia.

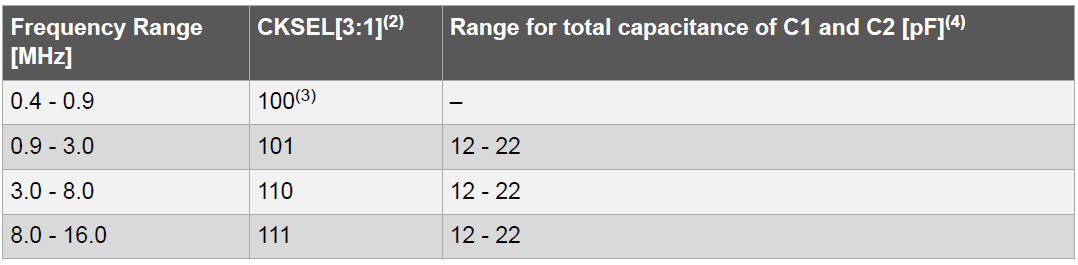
​Consulte [la sección 11 de la hoja de datos del dispositivo](http://ww1.microchip.com/downloads/en/DeviceDoc/40001906A.pdf), que especifica el número de ciclos de retardo CK para cada tipo de fuente de reloj y la configuración del fusible SUTx.

**Oscilador de cristal de baja potencia**

Los pines XTAL1 y XTAL2 son entrada y salida, respectivamente, de un amplificador inversor que se puede configurar para su uso como oscilador en chip, como se muestra en la figura a continuación. Se puede utilizar un cristal de cuarzo o un resonador de cerámica:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-xtal-connection.png)

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

[](https://microchipdeveloper.com/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 (consulte la sección 11.3 de 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 arranque son determinados por los fusibles SUT.

**Oscilador RC interno calibrado**

De forma predeterminada, el oscilador RC interno proporciona un reloj de 8,0 MHz. Aunque el voltaje y la temperatura dependen, este reloj puede ser calibrado con mucha precisión por el usuario. El dispositivo se envía con el fusible CKDIV8 programado, 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 a '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.

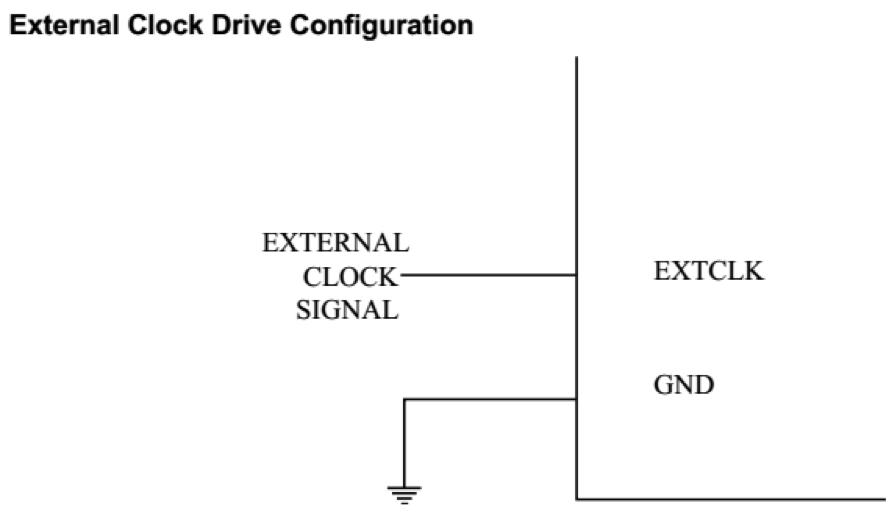
​Consulte [la nota de la aplicación AVR053](http://ww1.microchip.com/downloads/en/AppNotes/Atmel-2555-Internal-RC-Oscillator-Calibration-for-tinyAVR-and-megaAVR-Devices_ApplicationNote_AVR053.pdf), que describe el procedimiento para volver a calibrar el oscilador RC interno.

**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 el reloj del sistema programando los fusibles CKSEL a '0011'.

**Reloj externo**

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

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-ext-clk-connection.png)

**Búfer de salida de reloj**

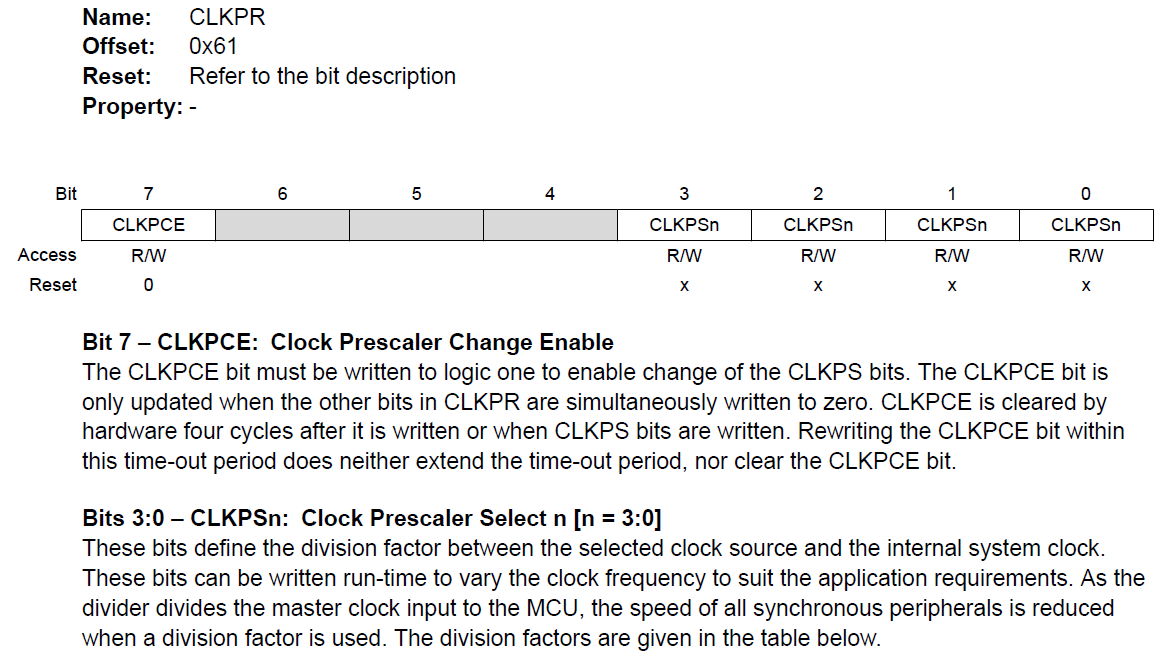
El dispositivo puede emitir el reloj del sistema en el pin CLKO. Para habilitar la salida, se debe programar el fusible CKOUT. Este modo es adecuado cuando el reloj del chip se utiliza para conducir otros circuitos en el sistema. El reloj también se emitirá durante el reinicio, y el funcionamiento normal 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, es el reloj del sistema dividido el que se emite.

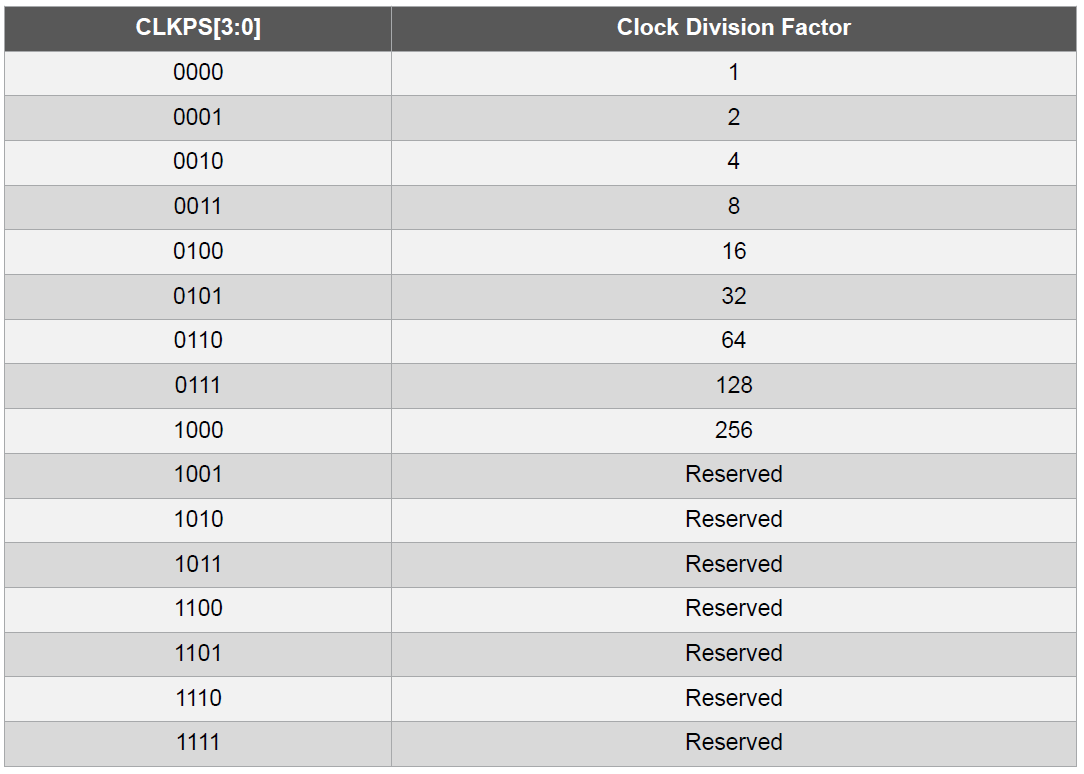
**Temporizador/Contador Oscilador**

El dispositivo utiliza el mismo oscilador de cristal para el oscilador de baja frecuencia y el oscilador de temporizador / contador. Consulte Oscilador de cristal de baja frecuencia para obtener detalles sobre el oscilador y los requisitos de cristal.  
  
En este dispositivo, los pines del temporizador/oscilador de contador (TOSC1 y TOSC2) se comparten con EXTCLK. Cuando se utiliza el temporizador / oscilador de contador, el reloj del sistema debe ser cuatro veces la frecuencia del oscilador. Debido a esto y al uso compartido de pines, el temporizador / oscilador de contador solo se puede usar cuando se selecciona el oscilador RC interno calibrado como fuente de reloj del sistema. La aplicación de una fuente de reloj externa a TOSC1 se puede realizar si el bit Habilitar entrada de reloj externo en el Registro de estado asincrónico (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 del reloj externo como entrada en lugar de un cristal de reloj de 32.768 kHz.

**Preescalador de 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 característica se puede utilizar para disminuir la frecuencia de 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. ClkE/SClkAdcClkCPU, y clkFLASH se dividen por un factor como se muestra en la descripción del CLKPR:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-clkpr-1.png)

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-overview/atmega328pb-clkpr-2.png)

**Escribiendo a CLKPR**

Al cambiar entre la configuración del preescalador, el preescalador del reloj del sistema garantiza que no se produzcan fallos en el sistema de reloj. También garantiza 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 ondulación que implementa el preescalador se ejecuta a la frecuencia del reloj indiviso, que puede ser más rápido que la frecuencia de reloj de la CPU. Por lo tanto, no es posible determinar el estado del preescalador: incluso si fuera legible, el tiempo exacto que lleva 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]), se tarda entre T1 + T2 y T1 + 2 \* T2 antes de que la nueva frecuencia de reloj esté activa. En este intervalo, se producen dos bordes de reloj activos. Aquí, T1 es el período de reloj anterior, y T2 es el período correspondiente a la nueva configuración del preescalador. Para evitar cambios involuntarios de frecuencia de 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 a cero: CLKPR = 0x80.
2. En cuatro ciclos, escriba el valor deseado en CLKPS[3:0] mientras escribe un cero en CLKPCE: CLKPR=0x0N

​Las interrupciones deben deshabilitarse al cambiar la configuración del preescalador para asegurarse de que no se interrumpe el procedimiento de escritura.

**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>  // Std integral type definitions

2 #include <avr/io.h>  // SFR definitions

3 #include <avr/interrupt.h> // ISR macros

4

5 void clkPrescaleSet(uint8\_t divisionFactor){

6    cli();                        // disable interrupts

7    CLKPR = (1 << CLKPCE);        // enable change of the CLKPSx bits

8    CLKPR = divisionFactor;       // update the CLKPSx bits

9    sei();                        // re-enable interrupts

10}

**Fusible CLKDIV8 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 bits CLKPS se restablecen a "0011", dando un factor de división de 8 en el arranque. 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 cualquier valor se puede escribir en los bits CLKPS independientemente de la configuración del fusible CKDIV8. El software de 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.

**Proyecto de ejemplo de oscilador megaAVR®**

**Objetivo**

Esta página proporciona un proyecto simple que demuestra el ajuste dinámico de la frecuencia del reloj del sistema a través de la modificación [del registro de preescalador del reloj del sistema (CLKPR)](https://microchipdeveloper.com/8avr:osc-mega-overview#system-clock-prescaler) en dispositivos megaAVR®. El ejemplo de código se ejecuta en la MCU [ATmega328PB](http://www.microchip.com/wwwproducts/en/ATmega328PB).  
  
La fuente de reloj del sistema se establece a través de bits de fusible de configuración para que sea el reloj externo de 16 MHz proporcionado por el chip mEDBG (consulte el diagrama de conexión a continuación).  
  
El proyecto configura el módulo Timer/Counter1 para que funcione en modo Clear-Timer-On-Compare (CTC) y, en una coincidencia de período, genera una interrupción de "tick" cada 100 mS. La fuente del reloj del temporizador está configurada para ser SYS\_CLK/64.  
  
El ISR timer/counter1 alterna LED0 e incrementa un contador. El bucle principal del programa supervisa el valor de Counter y actualiza el valor de CLKPR cada 10 segundos para cambiar dinámicamente la frecuencia SYS\_CLK.  
  
CLKPR se alterna entre la configuración div/1 y div/4, cambiando así el intervalo de alternancia (interrupción) de 100 ms a 400 mS respectivamente. Esto se puede ver como la velocidad de parpadeo LED0 cambia cada 10 segundos.

​Revise el archivo main.c del proyecto para obtener comentarios más detallados y una descripción de la operación.

**Materiales**

Herramientas de hardware

[ATmega328PB-XplainedMini-50px.png](http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)

ATmega328PB Xplained Mini  
Kit de evaluación

Herramientas de software

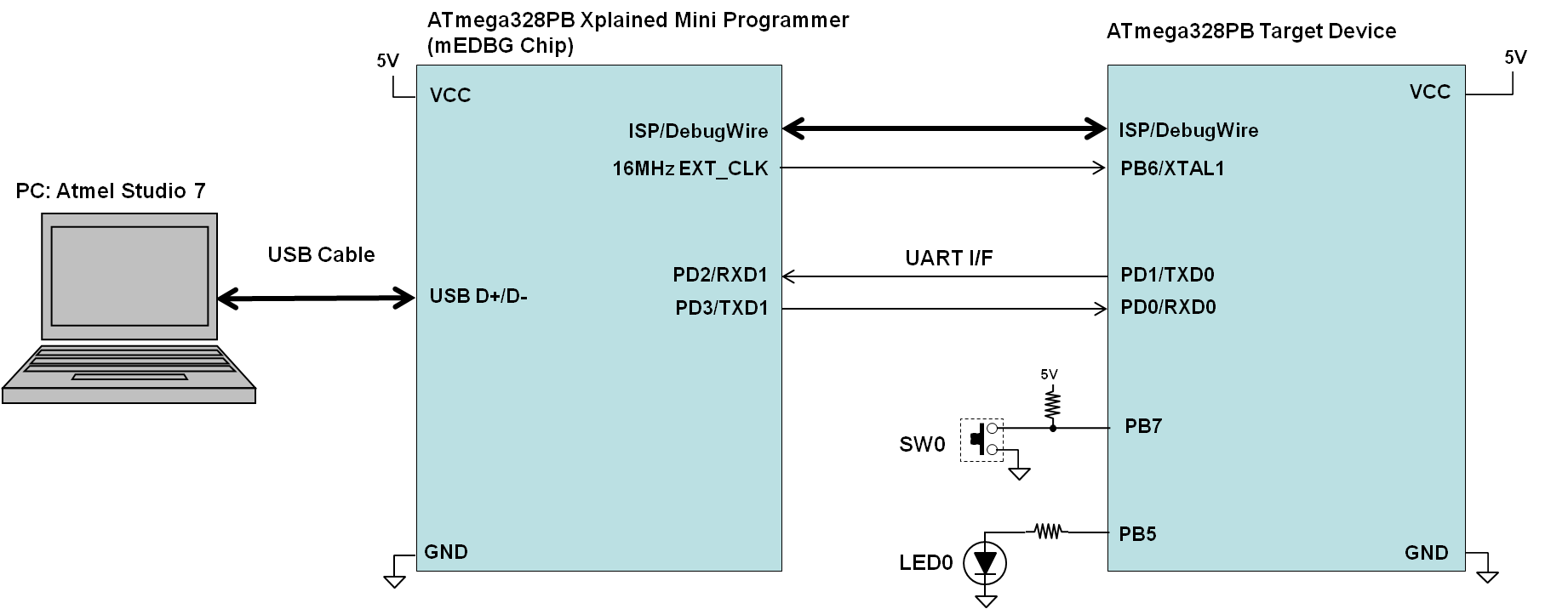
swtool-28px.png

Estudio Atmel®  
Entorno de desarrollo integrado

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

**Diagrama de conexión**

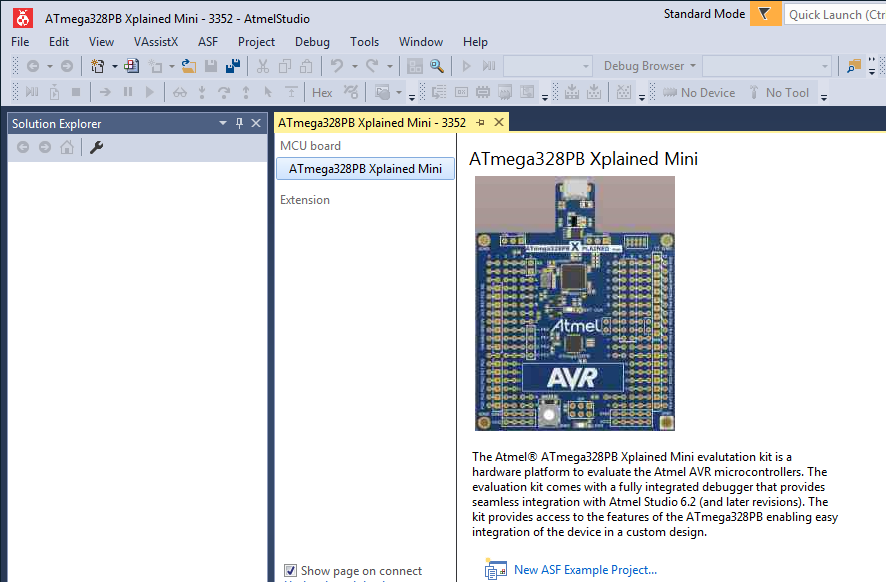
El chip mEDBG controla la interfaz de programación/depuración, además de suministrar un reloj de 16 MHz cuando la placa Xplained se conecta mediante un cable USB a un PC.

[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/xplained-mini-connection-diagram-as7.png)

​Los bits de fusible ATmega328PB CKSEL de destino son inicialmente inmutables en la placa Xplained Mini desde Atmel Studio. Los errores de "verificación" se mostrarán si se programan bits CLSEL con cualquier otra configuración que no sea "reloj externo".  
Esto se puede anular borrando el filtro de fusibles mEDBG como se describe en la sección 1.6.2 de la [Guía del usuario de ATmega328PB Xplained Mini](http://www.atmel.com/Images/Atmel-42469-ATmega328PB-Xplained-Mini_User-Guide.pdf).

**Procedimiento**

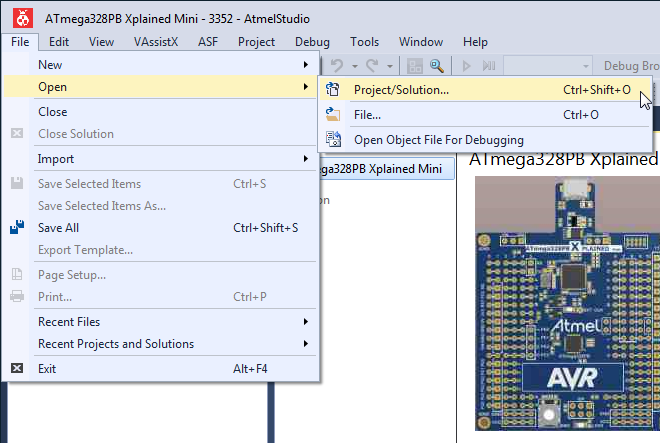
Conecte la placa ATmega328PB Xplained Mini a su computadora usando un cable USB-A-macho-a-Micro-B-macho. Inicie Atmel Studio 7. Si la placa se ha enumerado correctamente, debería ver la imagen de la placa aparecer en Studio como se muestra:

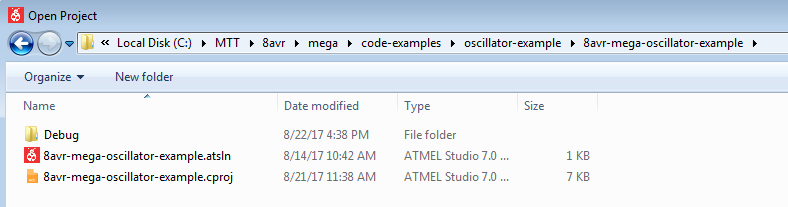
[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/xplained-mini-enumeration-success.png)

​El mini tablero Xplained se identifica por los últimos cuatro dígitos en su número de serie (ver pegatina en la parte inferior del tablero). En el ejemplo anterior, los últimos cuatro dígitos son "3352"

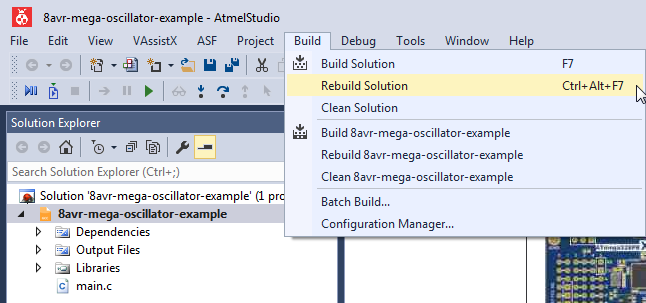
**1 Abrir la solución**

En Studio, seleccione **archivo > Abrir > proyecto/solución** y navegue hasta la ubicación guardada de la solución y, a continuación, seleccione el archivo 8avr-mega-oscillator-example.atsln:

[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/as7-open-solution.png)

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-example/as7-open-osc-solution-detail.png)

**2 Reconstruir la solución**

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-example/as7-rebuild-osc-solution.png)

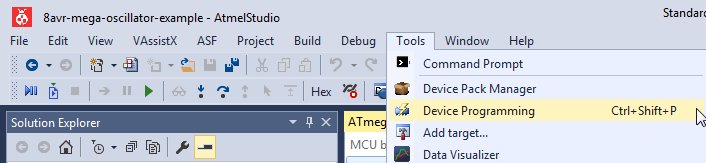
**3Programa los fusibles**

Hay varias opciones clave de configuración de hardware que deben configurarse. Los [siguientes ajustes de fusibles](https://microchipdeveloper.com/8avr:avrfuses) deben programarse en el dispositivo:

Ext: 0xFC

* Alto: 0xDF
* Bajo: 0xC0 (EXT CLK, aumento rápido de VDD, CLKDIV = 1)

Ingrese al cuadro de diálogo Programación de dispositivos como se muestra:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-example/as7-programming-fuses-osc-1.png)

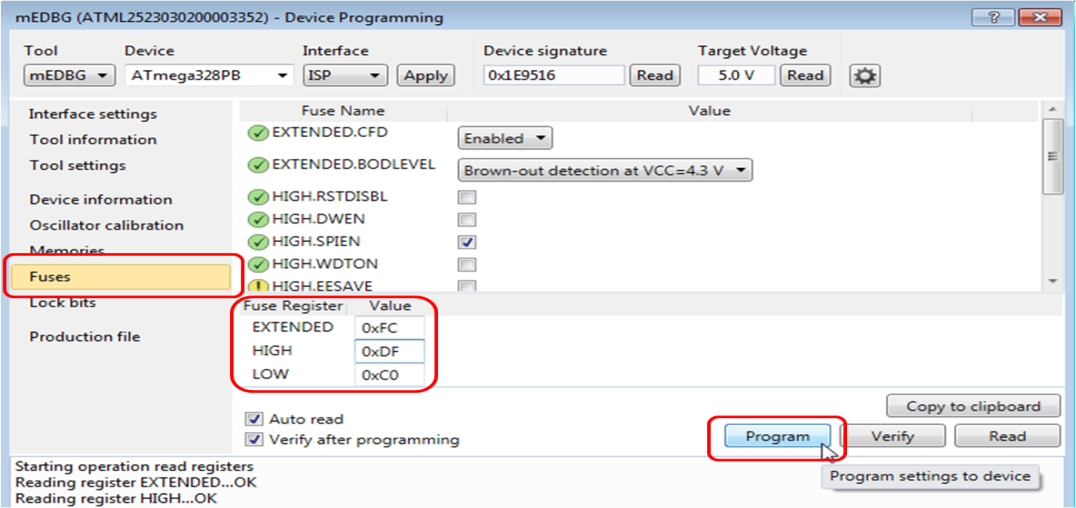
En el cuadro de diálogo Programación de dispositivos, seleccione Herramienta, Dispositivo e interfaz como se muestra y, a continuación, haga clic en Aplicar:

[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/as7-program-fuses-2.png)

Para verificar una conexión, seleccione Leer y compruebe que se encuentra una firma de dispositivo:

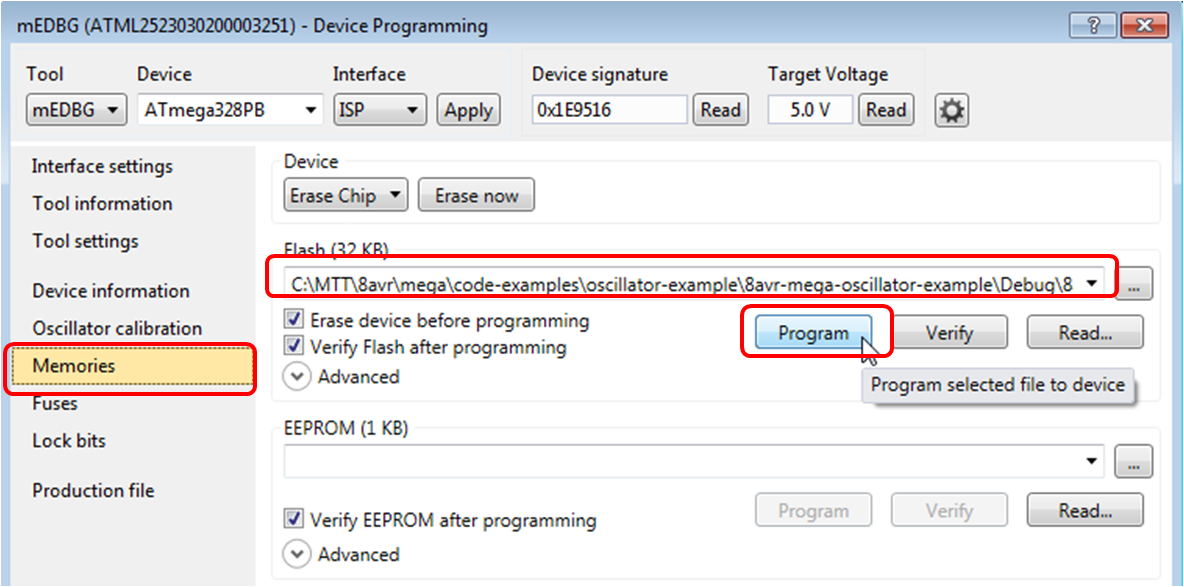
[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/as7-program-fuses-3.png)

Seleccione la subsección Fusibles, Introduzca los 3 valores de bytes de fusibles anteriores y, a continuación, haga clic en Programa como se muestra:

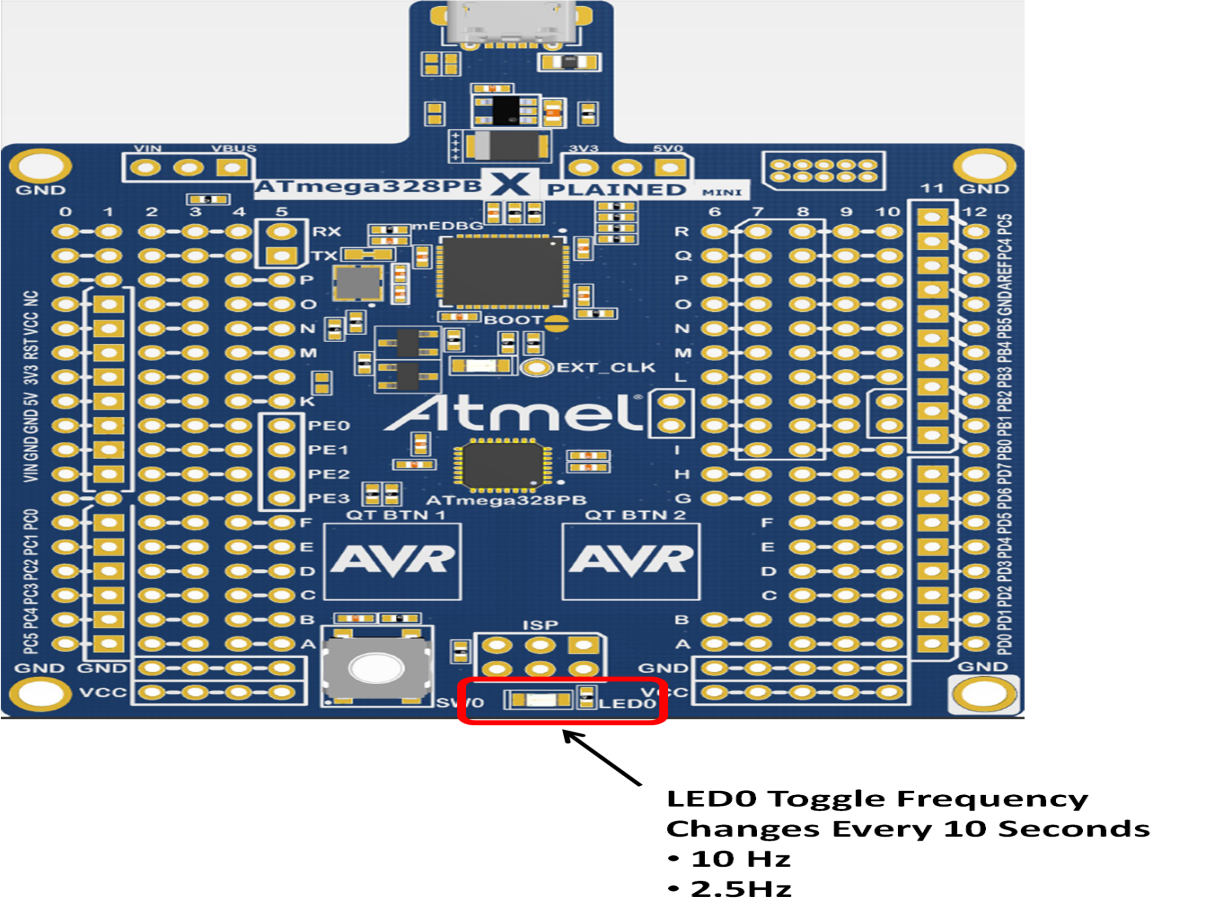
[](https://microchipdeveloper.com/local--files/8avr:interrupts-mega-example/as7-program-fuses-4.png)

**4Programar el archivo hexadecimal**

Mientras aún está en el cuadro de diálogo Programación de dispositivos, seleccione la subsección Memorias como se muestra. La ruta de acceso al archivo hexadecimal de la solución ya debería aparecer en el cuadro de diálogo. Haga clic en Programa como se muestra:

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-example/as7-program-hex-osc-example-1.png)

**Resultados**

[](https://microchipdeveloper.com/local--files/8avr:osc-mega-example/osc-mega-example-results.png)

Conclusiones

Este proyecto ha proporcionado un ejemplo de cómo ajustar dinámicamente la frecuencia del reloj del sistema en el MCU megaAVR®.

**USART (desarrollar carpeta)**

**Interrumpe (desarrollar carpeta)**

**Comparador analógico y referencia de voltaje (desarrollar carpeta)**

**Temporizador / Contadores (desarrollar carpeta)**

**Sensor de temperatura interno(desarrollar carpeta)**

**Operación de baja potencia(desarrollar carpeta)**

**Restablecer fuentes(desarrollar carpeta)**