

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 CKSEL Flash. Esta discusión es específica para el MCU 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.

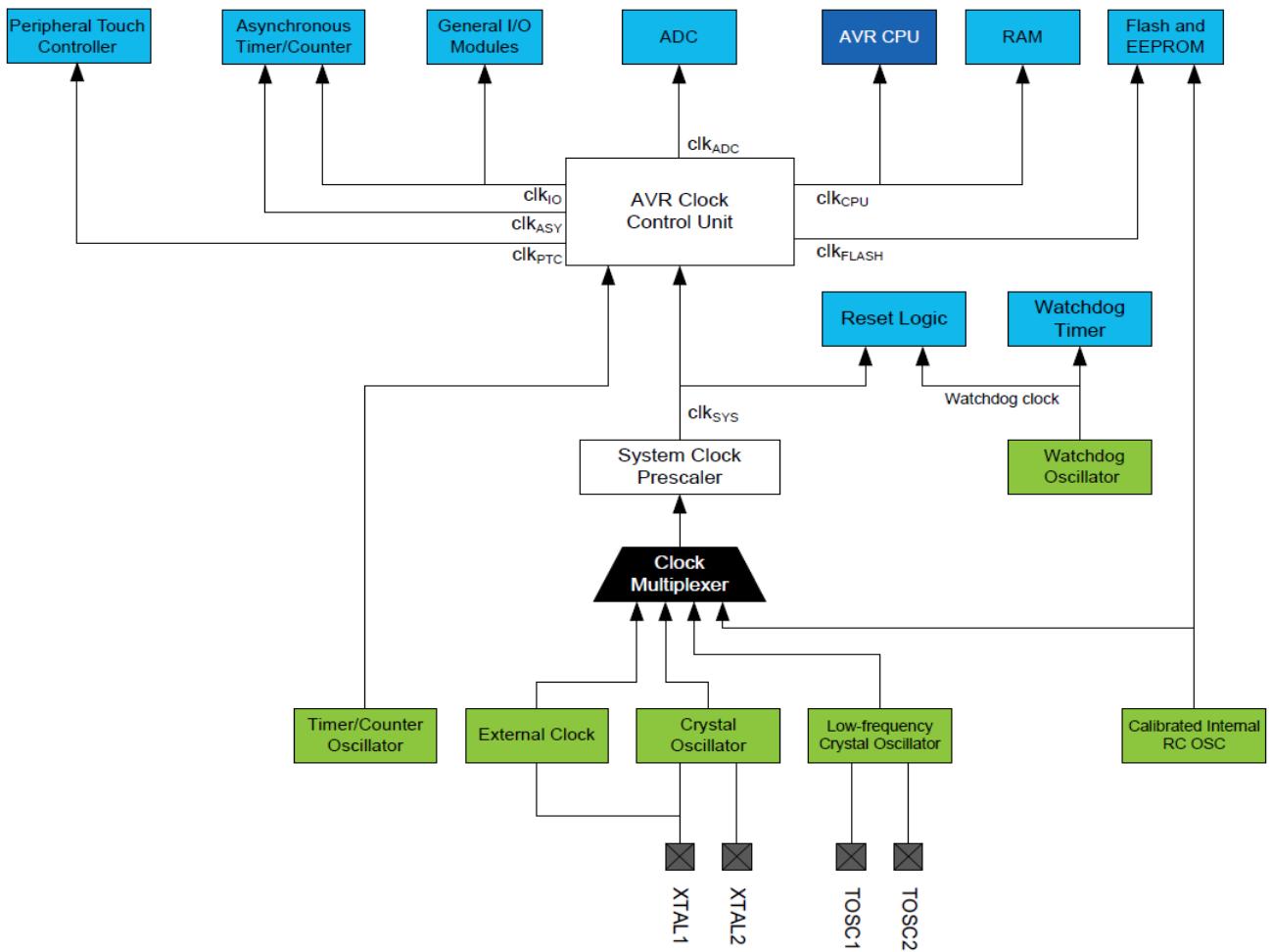
La frecuencia de reloj del sistema se puede cambiar durante el tiempo de ejecución escribiendo en el registro del preescalador de reloj del sistema (CLKPR).

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).

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. 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.



Fuentes de reloj

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.

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

Origen de reloj predeterminado

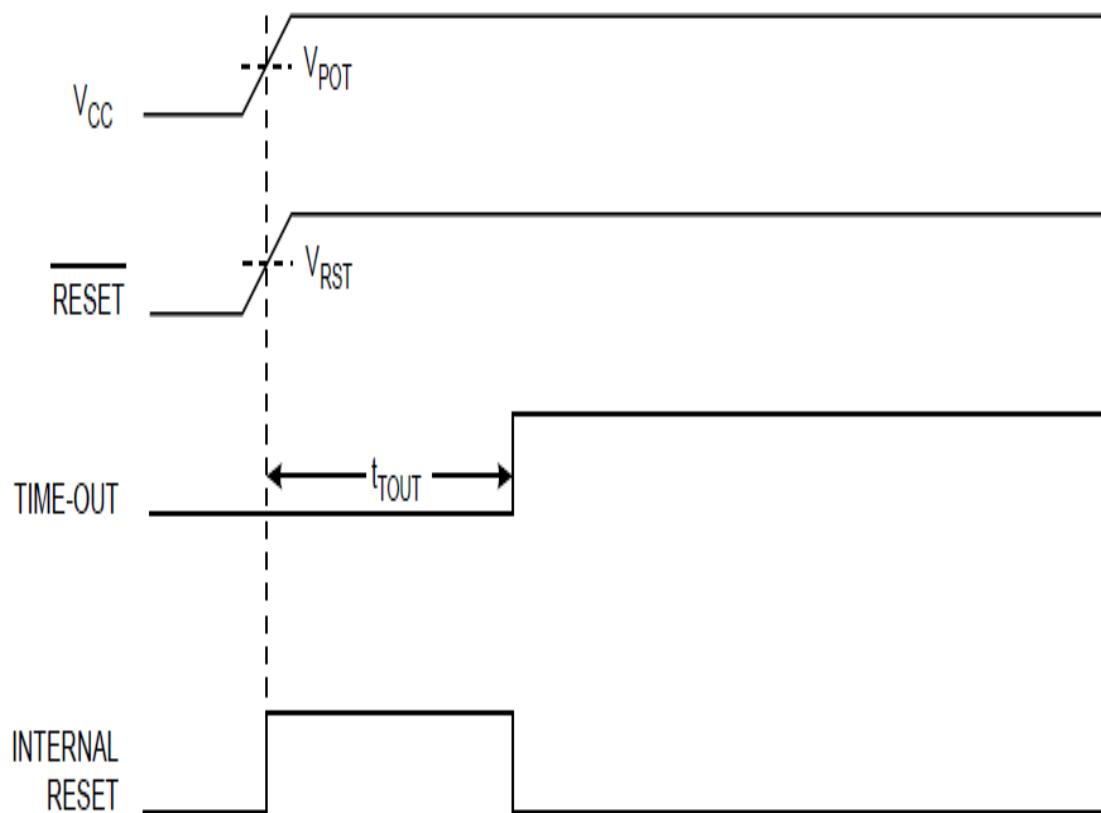
El dispositivo se envía con el oscilador RC interno seleccionado a 8.0 MHz y con el fusible CKDIV8 programado, lo que resulta en un reloj del sistema de 1.0 MHz. El tiempo de inicio se establece en máximo y el período de tiempo de espera está habilitado: CKSEL=0010, SUT=10, CKDIV8=0. Esta configuración 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_{CC} para empezar a oscilar y (ii) un número mínimo de ciclos oscilantes antes de que pueda considerarse estable.

Estabilidad de V_{CC}

Para garantizar una V_{CC} , el dispositivo emite un restablecimiento interno con un retraso de tiempo de espera (t_{TOUT}) después de que el restablecimiento del dispositivo sea liberado por todas las demás fuentes de restablecimiento:



El retraso (t_{out}) 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:

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

V_{CC} no se supervisa durante el retraso, por lo que es necesario seleccionar un retraso superior al V_{CC} 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 V_{CC} antes de que se libere el restablecimiento, y el retraso de tiempo de espera se puede deshabilitar. No se recomienda deshabilitar el retraso de tiempo de espera sin utilizar un circuito de detección de salida marrón.

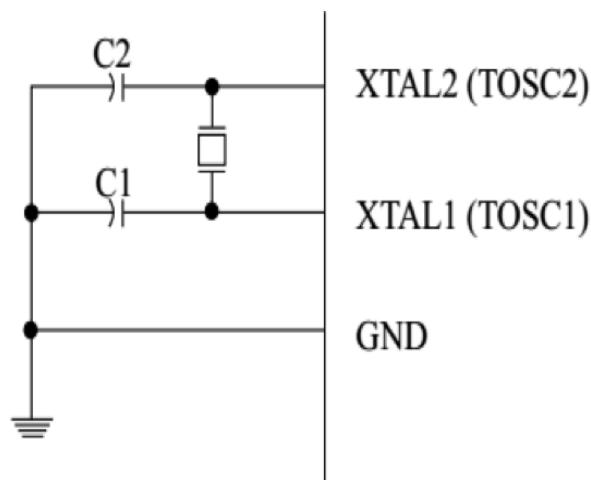
Estabilidad del oscilador

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, 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:



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:

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

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).

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, que describe el procedimiento para volver a calibrar el oscilador RC interno.

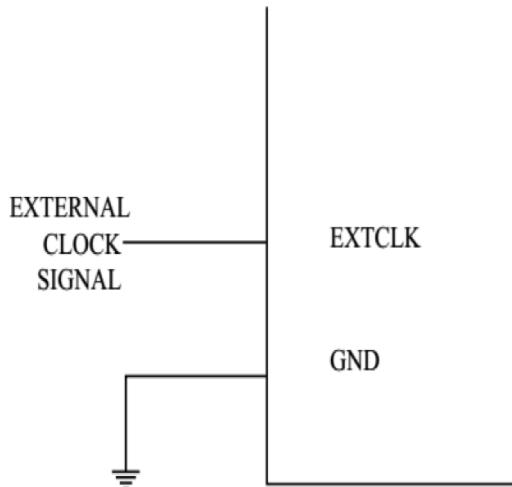
Oscilador interno de 128 kHz

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'.

External Clock Drive Configuration



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. $\text{Clk}_{\text{E/S}}$, Clk_{Adc} , Clk_{CPU} , y $\text{clk}_{\text{FLASH}}$ se dividen por un factor como se muestra en la descripción del 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.

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

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 interrumpe.

```
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) en dispositivos megaAVR®. El ejemplo de código se ejecuta en la MCU 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 Xplained Mini

Kit de evaluación

Herramientas de software



Estudio Atmel®

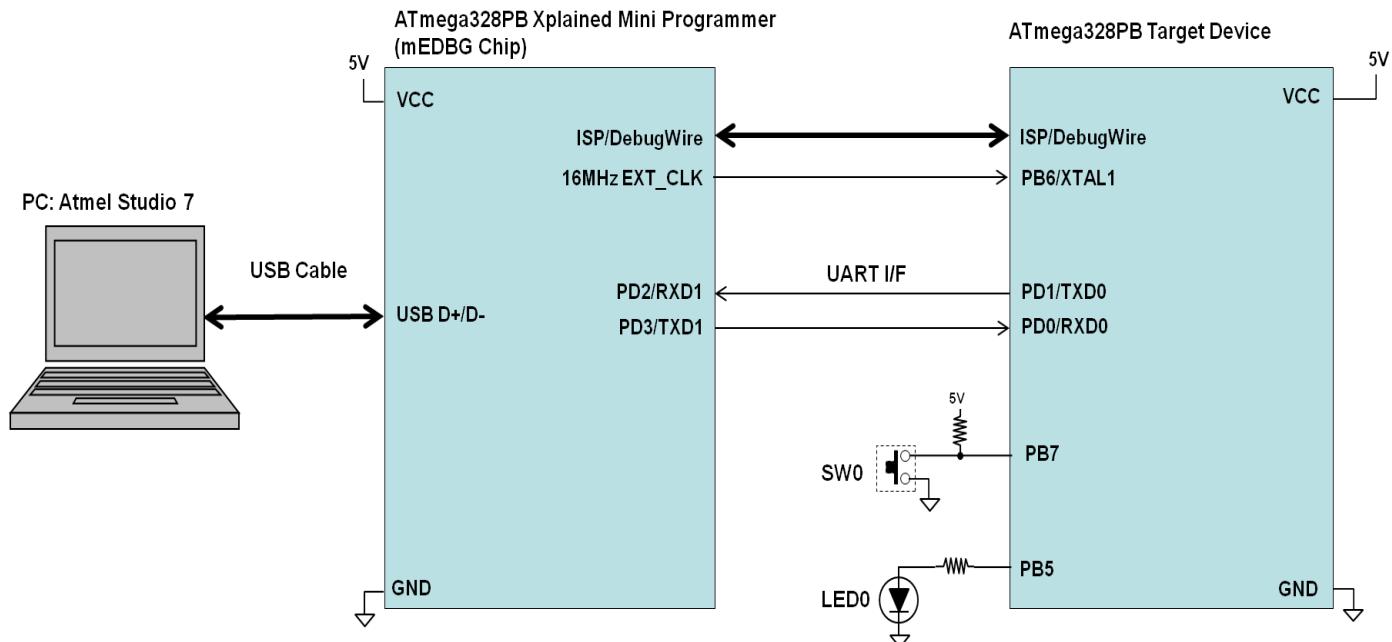
Entorno de desarrollo integrado

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

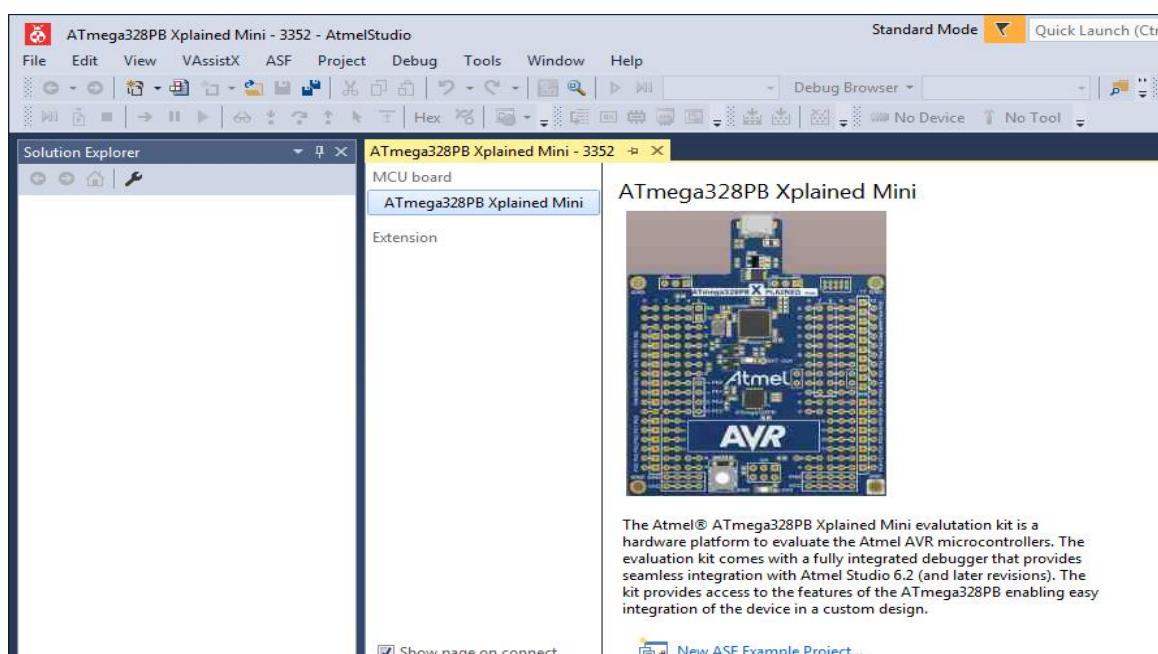
El chip mEDBG controla la interfaz de programación/depuración, además de suministrar un reloj de 16 MHz cuando la placa Xplained se conecta mediante un cable USB a un PC.



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.

Procedimiento

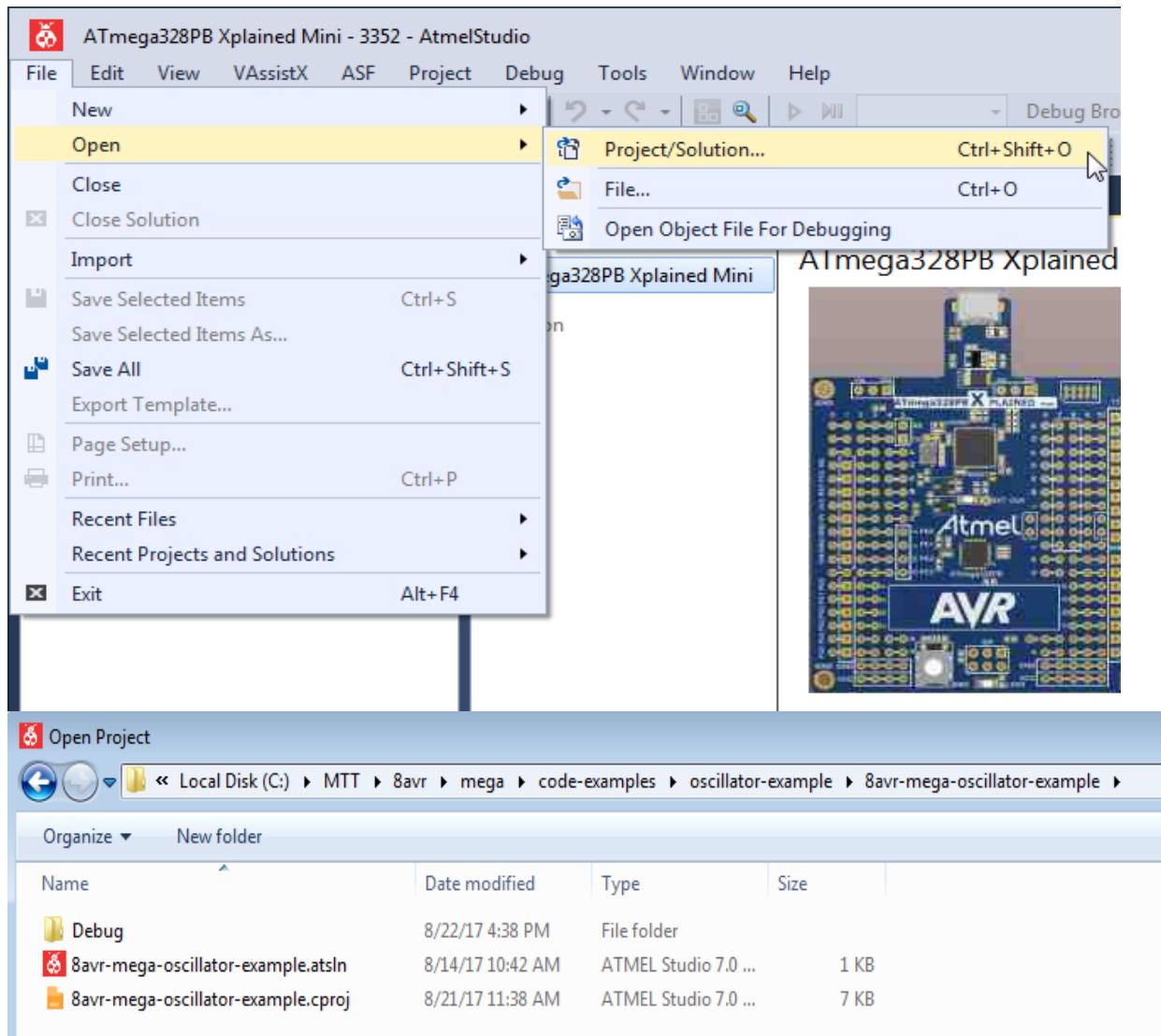
Conecte la placa ATmega328PB Xplained Mini a su computadora usando un cable USB-A-macho-a-Micro-B-macho. Inicie Atmel Studio 7. Si la placa se ha enumerado correctamente, debería ver la imagen de la placa aparecer en Studio como se muestra:



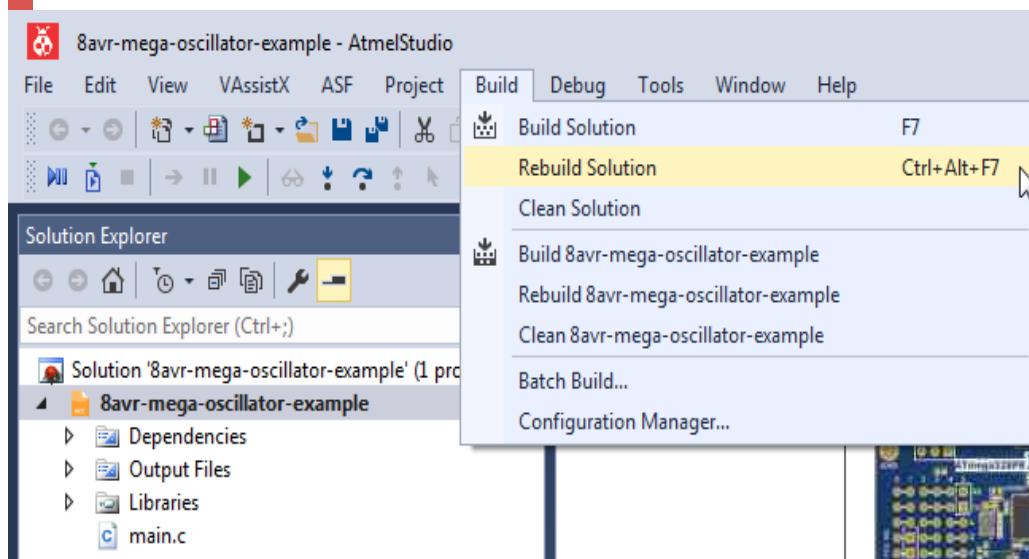
El mini tablero Xplained se identifica por los últimos cuatro dígitos en su número de serie (ver pegatina en la parte inferior del tablero). En el ejemplo anterior, los últimos cuatro dígitos son "3352"

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**:



2 Reconstruir la solución



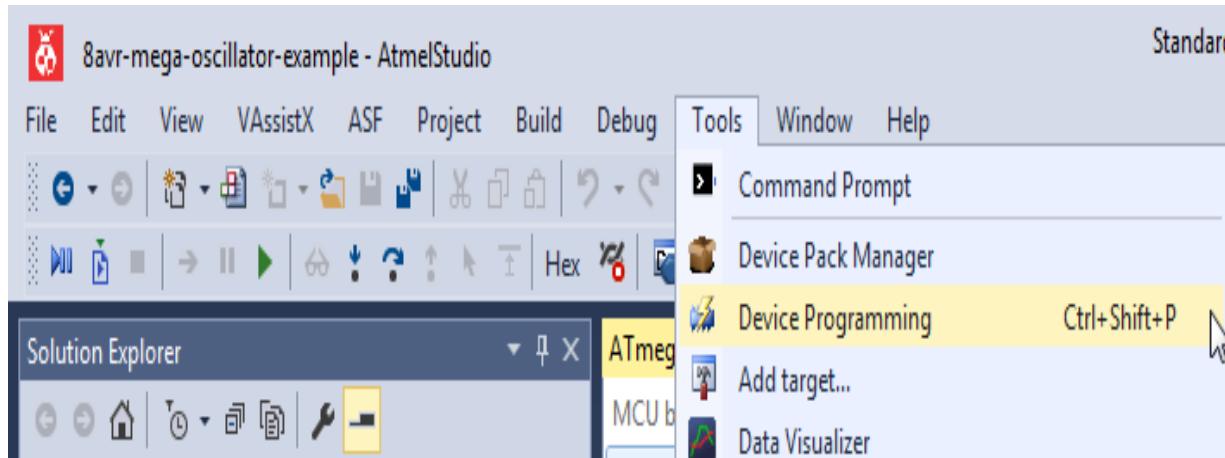
3 Programa los fusibles

Hay varias opciones clave de configuración de hardware que deben configurarse. Los siguientes ajustes de fusibles 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:



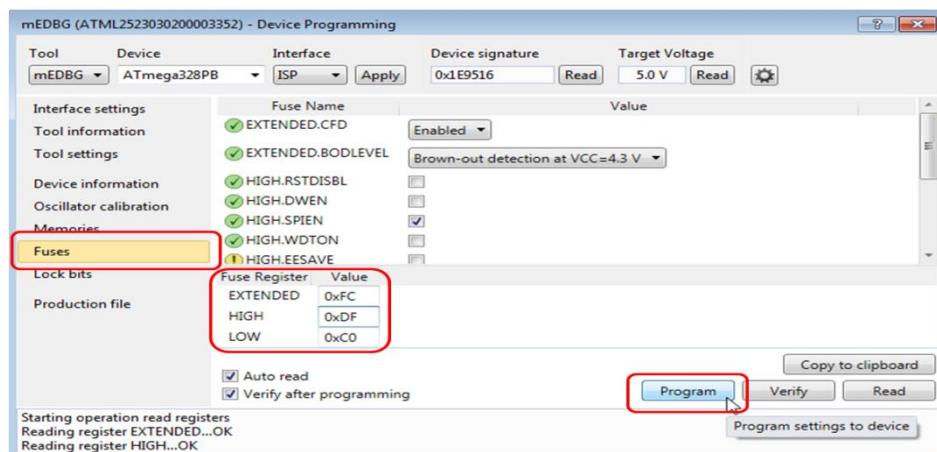
En el cuadro de diálogo Programación de dispositivos, seleccione Herramienta, Dispositivo e interfaz como se muestra y, a continuación, haga clic en Aplicar:



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

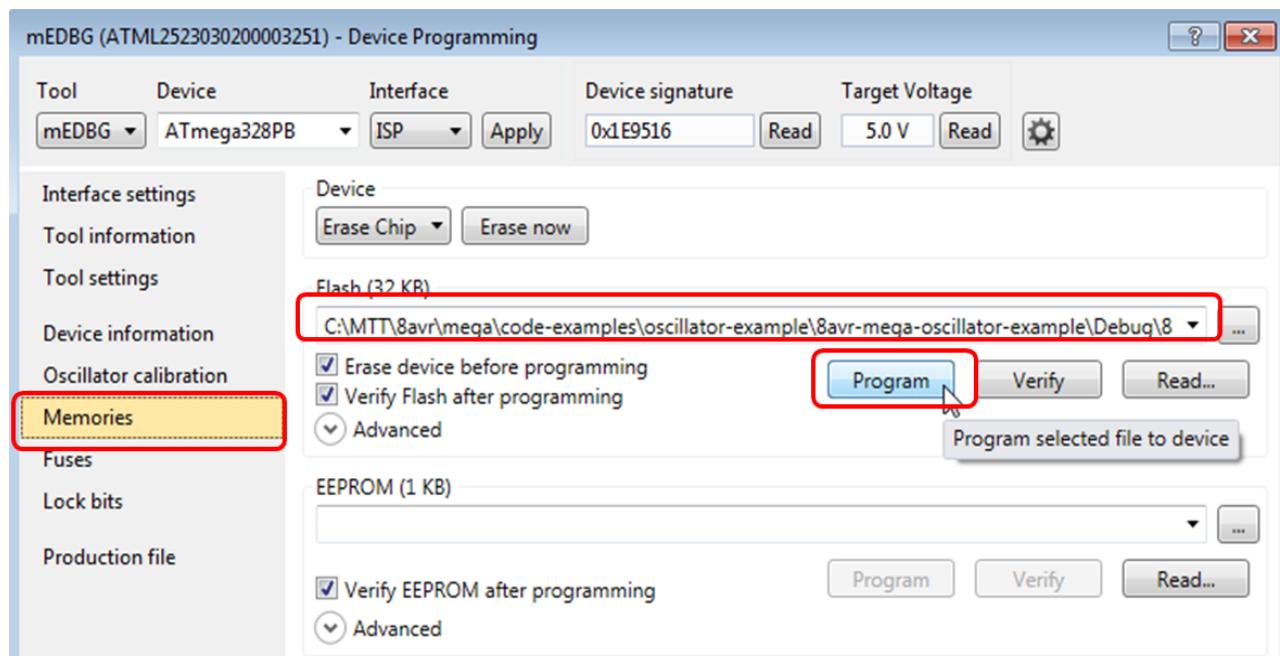


Seleccione la subsección Fusibles, Introduzca los 3 valores de bytes de fusibles anteriores y, a continuación, haga clic en Programa como se muestra:

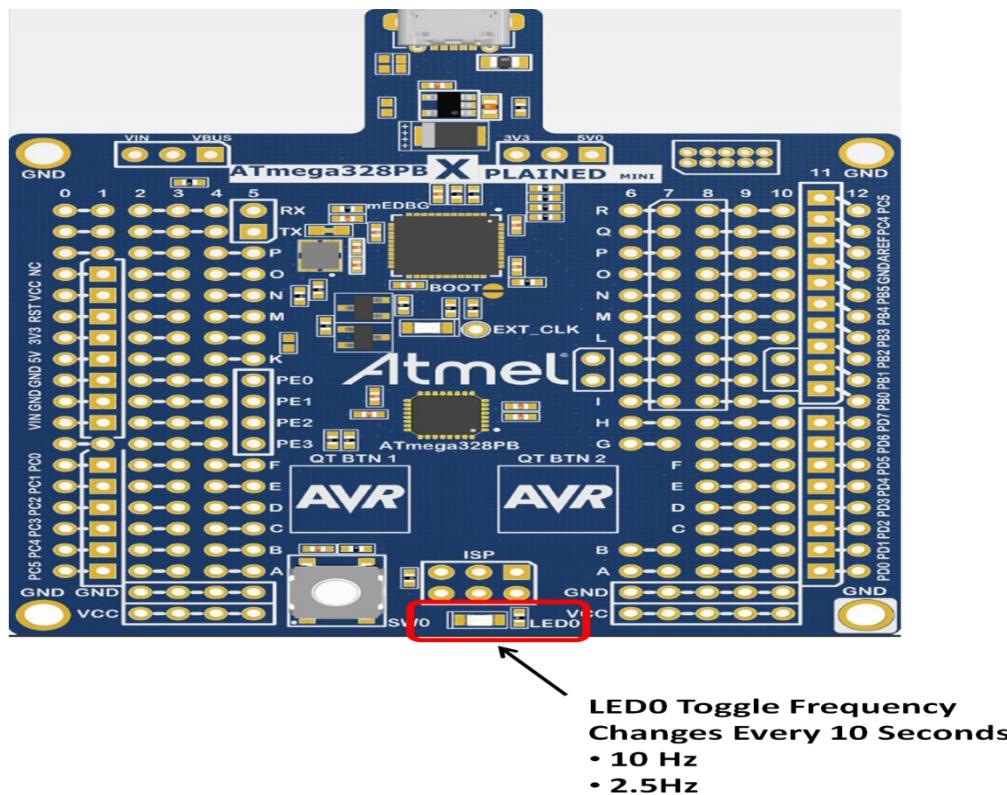


4 Programar el archivo hexadecimal

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:



Resultados



Conclusiones

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

USART

Descripción general de AVR® USART

Los microcontroladores Microchip AVR® de 8 bits contienen un periférico de comunicación altamente flexible conocido como USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter).

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 los dos modos principales: síncrono o asíncrono. Este módulo se centra en el modo de operación asíncrono.

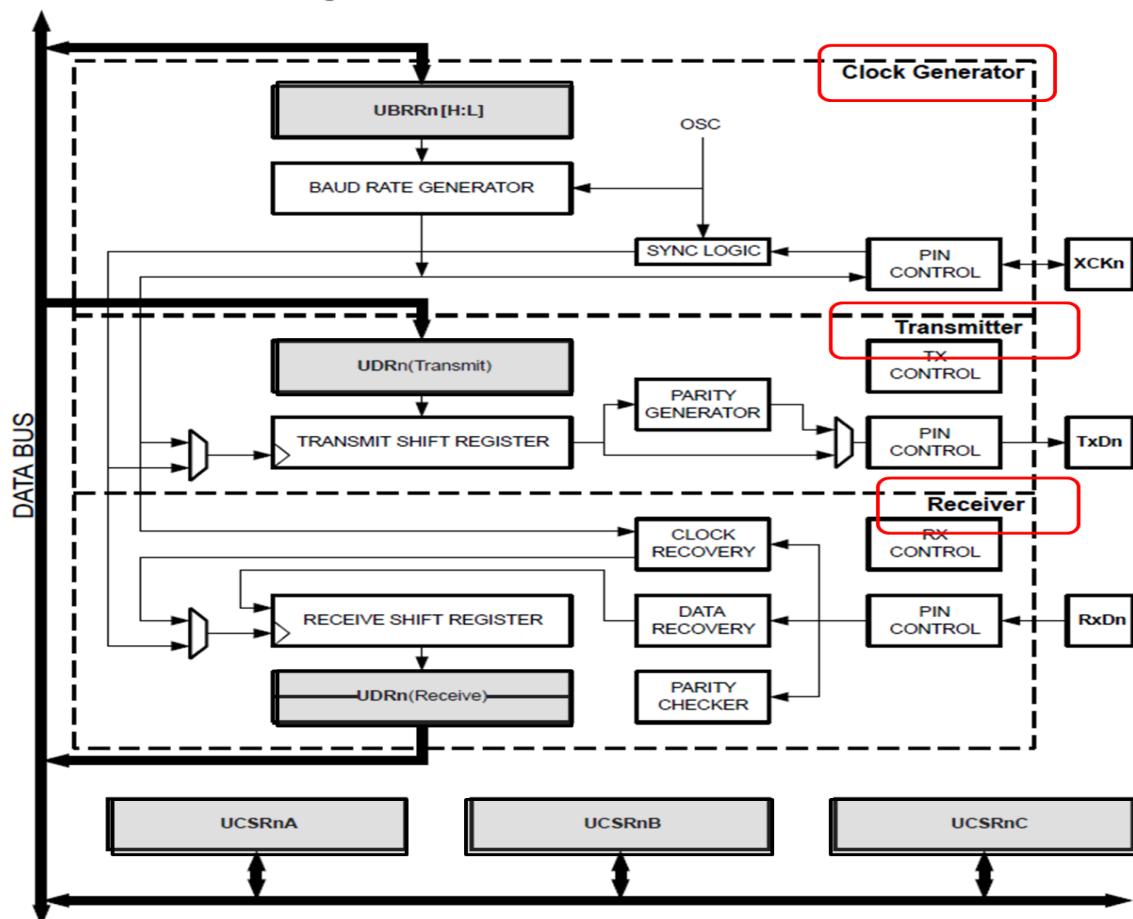
Configuración de megaAVR® USART

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.

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



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.
- El control de velocidad en baudios registra UBRRn[H:L] utilizado por el generador de reloj.

"n" en el nombre de 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

Uso del USART (Resumen)

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

1. Elija una velocidad en baudios y programe los registros UBRRn[H:L] en consecuencia.
2. Habilite las secciones de transmisión y recepción serie usart.
3. Si está transmitiendo, espere hasta que el registro de desplazamiento de transmisión esté vacío (sondeo en UCSRnA.UDREn) y, a continuación, cargue el byte de datos en UDRn.
4. Si recibe, espere hasta que se establezca el bit de recepción de datos del receptor (sondeo en UCSRnA.RXCn) y, a continuación, lea los datos de UDRn. La lectura de UDRn borra automáticamente el bit y prepara el hardware para recibir el siguiente byte.

Inicialización

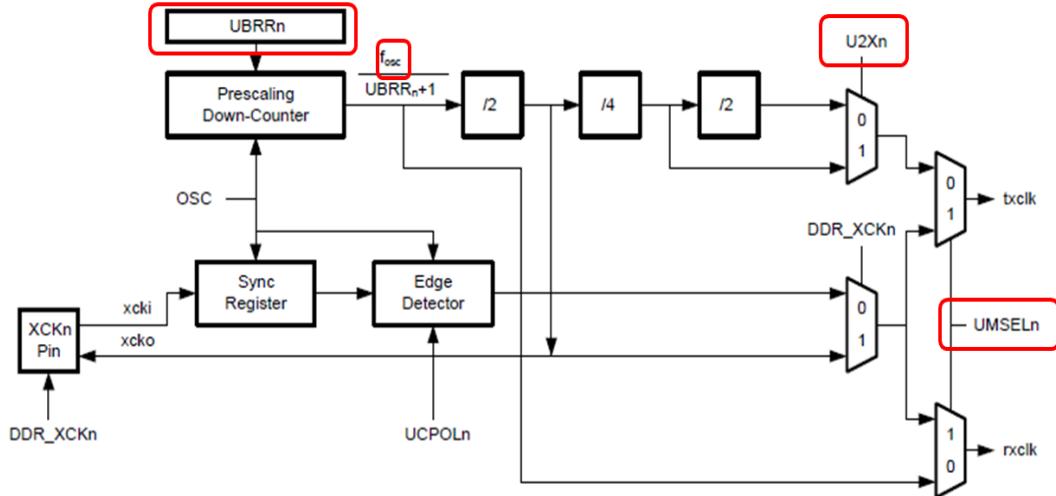
El USART debe inicializarse antes de que pueda tener lugar cualquier comunicación. El proceso de inicialización normalmente consiste en:

- Ajuste de la velocidad en baudios,
- Configuración del formato de marco y
- Habilitación del Transmisor o del Receptor dependiendo del uso.

Configuración de la velocidad en baudios

La generación de reloj interno se utiliza para el modo de operación asíncrono. La lógica de generación de reloj genera el reloj base para el transmisor y el receptor (se resaltan los registros de claves y los bits de control):

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.**

Selección de modo USART (UMSELn)

La ecuación de velocidad en baudios utilizada por el módulo se establece en función del modo de funcionamiento. Para la operación en modo asincrónico, los bits usart mode Select en el registro de control y estado USART C (UCSRnC.UMSELn[1:0]) se utilizan para seleccionar la operación asincrónica (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

Modo de doble velocidad (U2Xn)

Para el modo asíncrono, la velocidad USART TX se puede duplicar estableciendo el bit U2Xn en el registro UCSRnA (UCSRnA.U2Xn = 1).

Con el modo de doble velocidad establecido, el receptor solo usará la mitad del número de muestras (reducido 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 más precisa y un reloj del sistema cuando se utiliza este modo.

Registro de tarifas en baudios (UBRRn)

El REGISTRO DE VELOCIDAD EN BAUDIOS USART (UBRRn) y el contador descendente conectado a él funcionan como un preescalador programable o generador de velocidad en baudios. El contador descendente, funcionando al reloj del sistema (f_{osc}), se carga con el valor UBRRn cada vez que el contador ha contado hasta 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, dependiendo del 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 utilizan una máquina de estado que utiliza 2, 8 o 16 estados dependiendo del modo establecido por el estado de los bits UMSEL, U2Xn y DDR_XCK.

La siguiente tabla contiene ecuaciones para calcular la velocidad en baudios (en bits por segundo) y para calcular el valor UBRRn 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)	$BAUD = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$

- BAUD: Velocidad en baudios (en bits por segundo, bps)
- f_{osc} : Frecuencia de reloj del oscilador del sistema
- UBRRn: Contenidos de los Registros UBRRnH y UBRRnL, (0-4095).

La biblioteca AVR-LIBC Setbaud contiene macros útiles para calcular los valores correctos para escribir en registros UBRRnH y UBRRnL. Consulte el ejemplo de código de inicialización a continuación.

Las tablas también se proporcionan en la hoja de datos del dispositivo que contiene valores UBRRn para velocidades de baudios comunes, dadas varias frecuencias de oscilador:

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

Baud Rate [bps]	$f_{osc} = 16.0000MHz$				$f_{osc} = 18.4320MHz$				$f_{osc} = 20.0000MHz$			
	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%

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 fotograma

USART Control and Status Register C (UCSRnC) se utiliza para configurar el formato de trama de comunicación UART: paridad, número de bits de parada y número de bits de datos. La configuración para el formato de marco típico "8N1" es la siguiente:

- UPM[1:0] = 00 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	0	0	0	0	0	1	1	0

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	0	0	0	0	0	0	0	0

Cuando el transmisor está habilitado, el usart reemplaza el funcionamiento normal del puerto del pin TxDn y se le da la función de salida serie del transmisor.

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

Habilitación del receptor

El receptor USART se habilita escribiendo el bit Receive Enable (RXEN) en el registro UCSRnB en '1':

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	0	0	0	0	0	0	0	0

Cuando el Receptor está habilitado, el funcionamiento normal del puerto del pin RxDn es anulado por el USART y se le asigna la función como entrada serie del Receptor.

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

Ejemplo de código

En el siguiente ejemplo de código de inicialización USART se utiliza la biblioteca de utilidades setbaud en AVR-LIBC. Esta biblioteca proporciona macros que utilizan 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 será de forma predeterminada +/- 2%.

Salidas

Suponiendo que el BAUD solicitado es válido para el F_CPU dado, la UBRR_VALUE de macros se establece en el valor de preescalador requerido. Se proporcionan dos macros adicionales para los bytes bajos y altos del preescalador, respectivamente: UBRRL_VALUE se establece en el byte inferior del UBRR_VALUE y UBRRH_VALUE se establece en el byte superior. Se definirá una USE_2X de macro adicional. Su valor se establece en 1 si la velocidad BAUD deseada dentro de la tolerancia dada solo se puede lograr estableciendo el bit U2Xn en la configuración UART. Se definirá a 0 si no se necesita U2Xn.

```
1 #define F_CPU 16000000UL           // required for setbaud & other libraries
2 #define BAUD 38400UL              // desired baud
3 #define BAUD_TOL 2                // desired baud rate tolerance (+/- %)
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 = UBRR0L_VALUE;
14    #if USE_2X                      // USE_2X defined by setbaud.h based on inputs
15    UCSR0A |= (1 << U2X0);
16    #else
17    UCSR0A &= ~(1 << U2X0);
18    #endif
19
20    // Set the Mode & Frame Parameters
21
22    UCSR0C = 0x06;                  // Asynchronous, 8-data, No parity, 1-stop
23
24    // Enable USART0 Transmitter and Receiver
25
26    UCSR0B = (1 << TXEN0) | (1 << RXEN0);
27
28 }
```

La biblioteca setbaud genera mensajes de advertencia durante la compilación si los parámetros de entrada generan un ajuste de velocidad BAUD que producirá una velocidad en baudios fuera del BAUD_TOL deseado.

Comunicaciones de datos

Transmitir

Una transmisión de datos se inicia cargando el búfer de transmisión con los datos que se van a transmitir. La CPU puede cargar el búfer de transmisión escribiendo en el registro UDRn. Para el funcionamiento sondeado, el firmware debe supervisar el indicador de registro de datos vacío (UCSRnA.UDREn) antes de cargar UDRn.

Los datos almacenados en búfer en el búfer de transmisión se moverán al Registro de turnos cuando el Registro de turnos esté listo para enviar una nueva trama. El registro de turnos 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 turnos se carga con nuevos datos, transferirá un fotograma completo a la velocidad dada por el Registro 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

El indicador de interrupción completa de transmisión (USCRnA.TXCn) está establecido y se puede generar una interrupción TX opcional (si está habilitada) cuando se ha desplazado toda la trama en el registro de cambios. El bit de indicador USCRnA.TXCn se borra automáticamente cuando se ejecuta una interrupción completa de transmisión, o se puede borrar escribiendo 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 cambio 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.

Se establece el indicador de interrupción completa de recepción (RXCn) y se puede generar una interrupción RX opcional (si está habilitada) cuando toda la trama del 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 APIs 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, sends the data
7     UDR0 = data;
8
9 }
10
11 unsigned char USART0_Receive(void) {
12
13     // Wait for data to be received
14     while(!(UCSR0A & (1 << RXC0)));
15
16     // Get and return received data from buffer
17     return UDR0;
18
19 }
```

MegaAVR® USART Ejemplo (Sondeado)

Objetivo

Esta página proporciona un proyecto simple que demuestra el funcionamiento sondeado del periférico USART en dispositivos megaAVR®. El ejemplo de código se ejecuta en la MCU ATmega328PB.

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. El bucle principal monitorea estas señales de tick para implementar un reloj de hora del día (formato HH: MM: SS). LED0 también se alterna en cada evento de tick.

La pantalla del reloj se envía a USART0 TX cada segundo y se controla mediante los caracteres de "control" de entrada del usuario recibidos en USART0 RX:

- 'u' permite actualizaciones en la pantalla del reloj cada segundo
- 'f' bloquea las actualizaciones de la pantalla

La interfaz de reloj se muestra utilizando un programa de emulador de terminal, como Tera Term.

Materiales

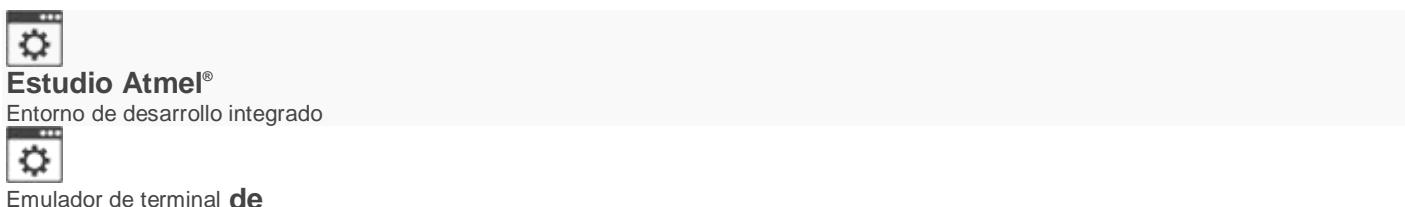
Herramientas de hardware



ATmega328PB Xplained Mini

Kit de evaluación

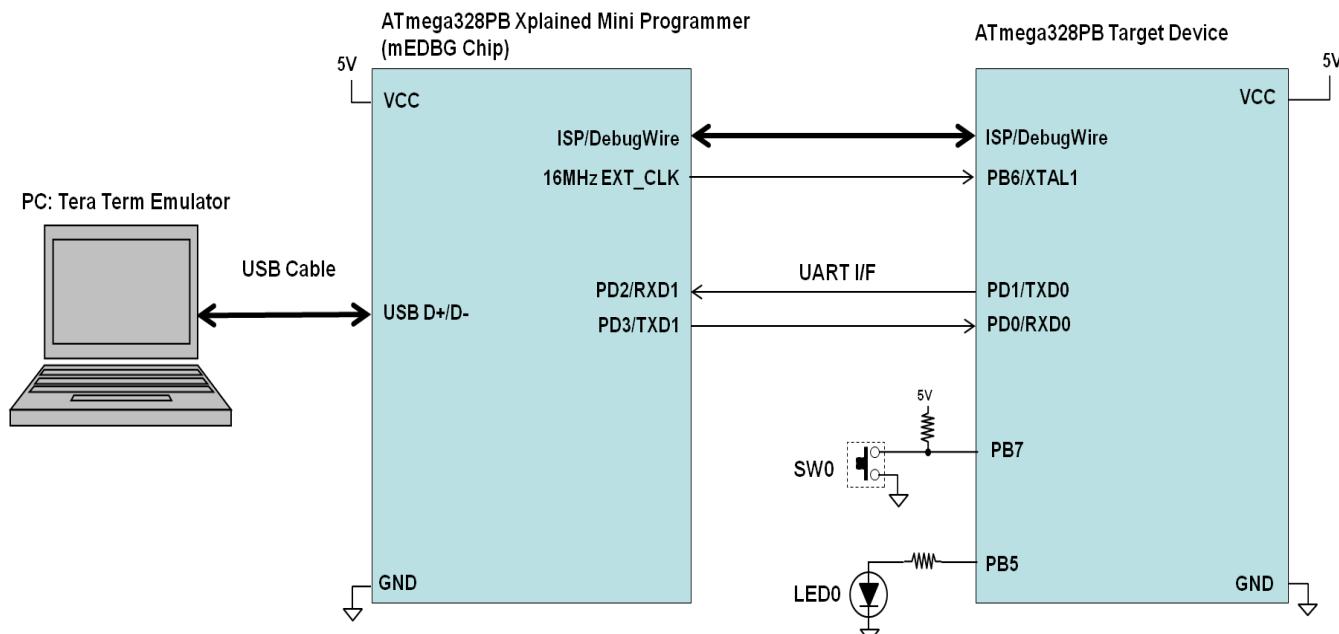
Herramientas de software



Debería ver la carpeta C:\MTT\8avr\mega\code-examples\usart-example-polled\8avr-mega-usart-example-polled que contiene la solución 8avr-mega-usart-example-polled.atsln

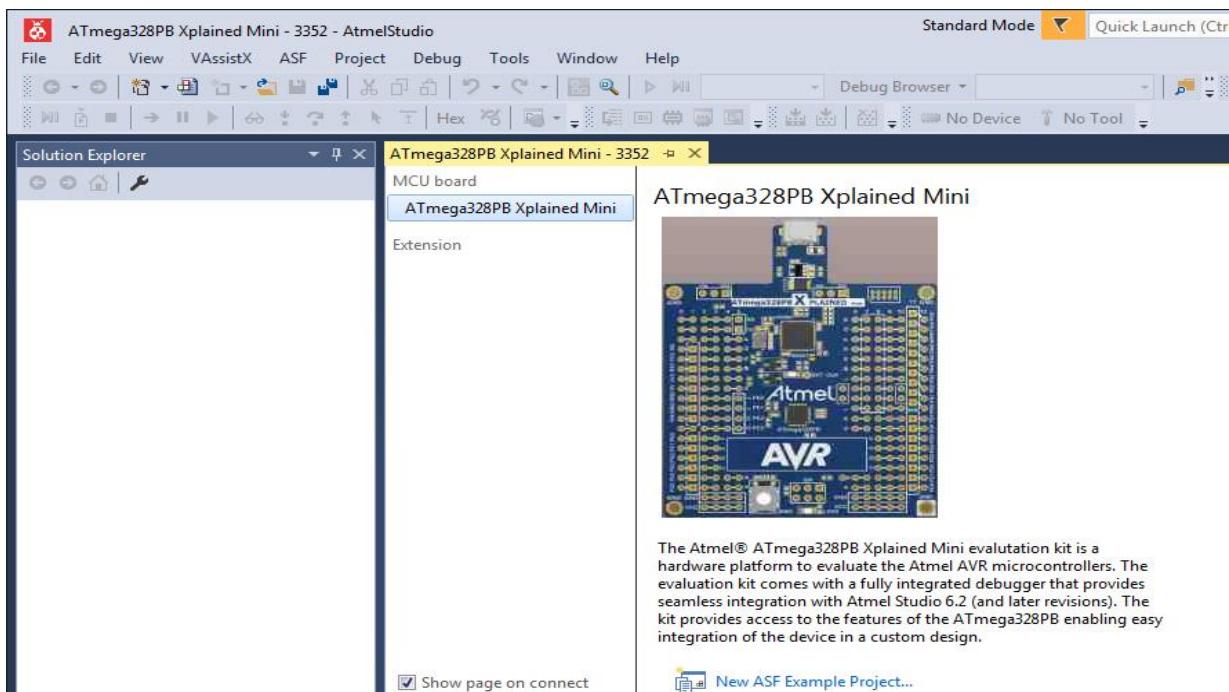
Diagrama de conexión

El módulo USART0 del dispositivo ATmega328PB de destino está conectado a la interfaz USART del chip mEDBG. El chip mEDBG realiza la conversión serie USB enumerando como un puerto COM virtual de clase CDC en la PC y presentando los datos USART de destino en esta interfaz. Tenga en cuenta que el mEDBG también controla la interfaz de programación / depuración, además de suministrar un reloj de 16MHz cuando la placa Xplained se conecta a través de un cable USB a una PC.



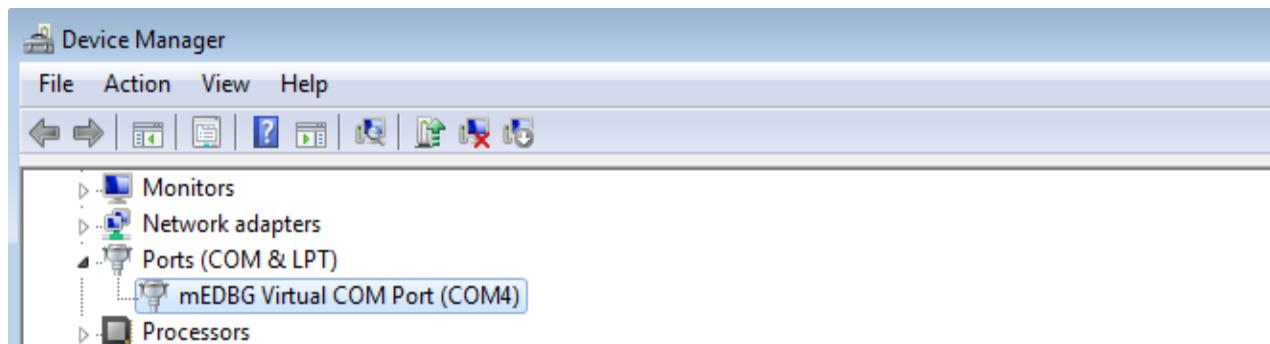
Procedimiento

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:



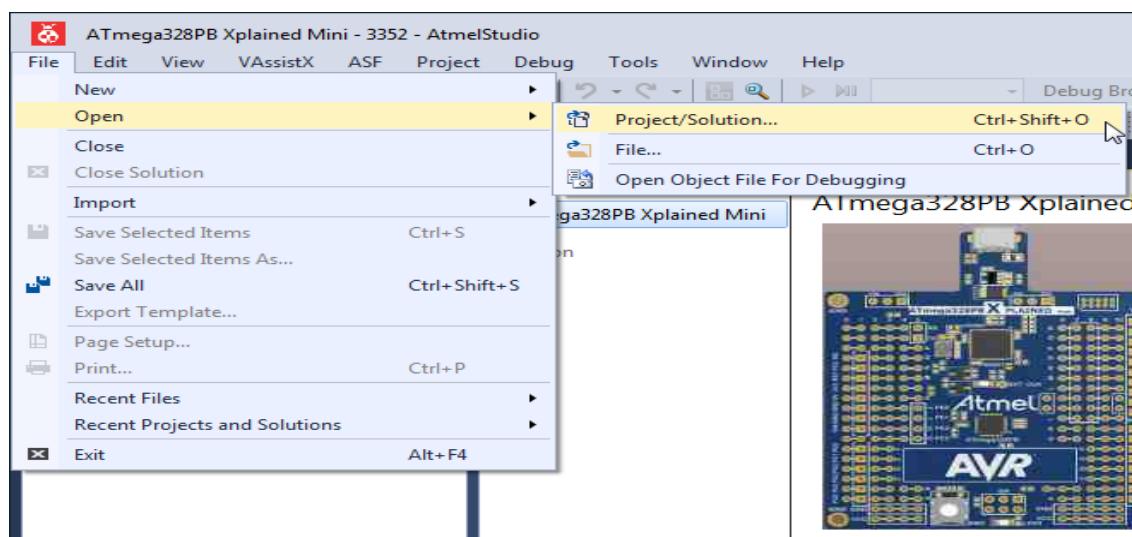
El mini tablero Xplained se identifica por los últimos cuatro dígitos en su número de serie (ver pegatina en la parte inferior del tablero). En el ejemplo anterior, los últimos cuatro dígitos son "3352"

También debería ver un puerto COM virtual mEDBG enumerado en el visor del Administrador de dispositivos de Windows. Tenga en cuenta el número de puerto COM asignado a su placa:



1 Abrir la solución

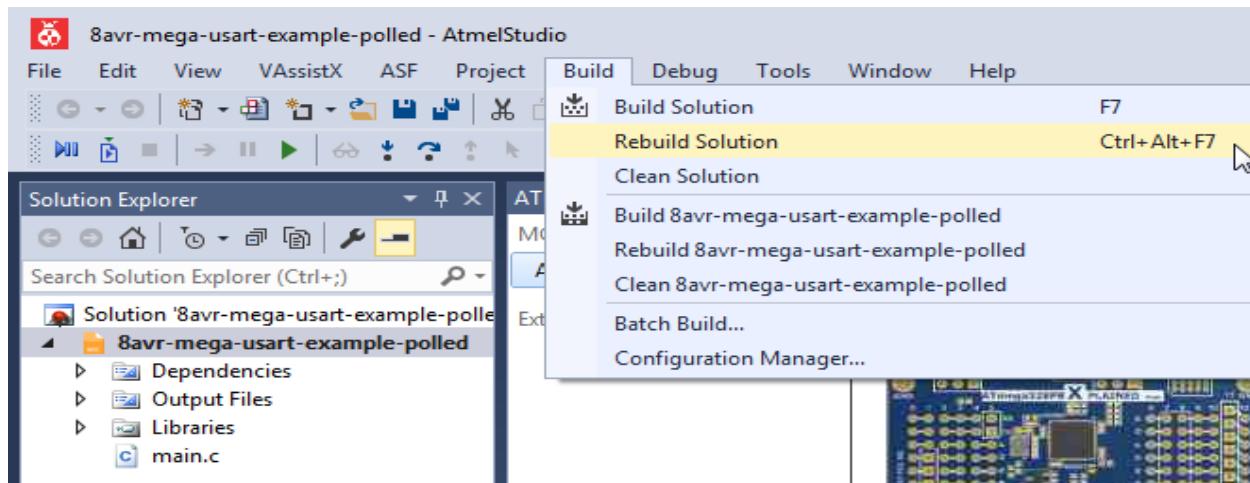
En Studio, seleccione Archivo » Abrir » Proyecto/Solución y navegue hasta la ubicación guardada de la solución:



Open Project				
Computer > Local Disk (C:) > MTT > 8avr > mega > code-examples > usart-example-polled > 8avr-mega-usart-example-polled				
Organize		New folder		
Name	Date modified	Type	Size	
Debug	6/12/17 12:47 PM	File folder		
8avr-mega-usart-example-polled.atsln	6/8/17 3:52 PM	ATMEL Studio 7.0 ...	1 KB	
8avr-mega-usart-example-polled.cproj	6/12/17 12:25 PM	ATMEL Studio 7.0 ...	8 KB	

Para revisar los procedimientos para configurar/usuarios el megaAVR® USART (tal como se implementa en el archivo main.c del proyecto), revise la página de configuración de megaAVR® USART.

2 Reconstruir la solución

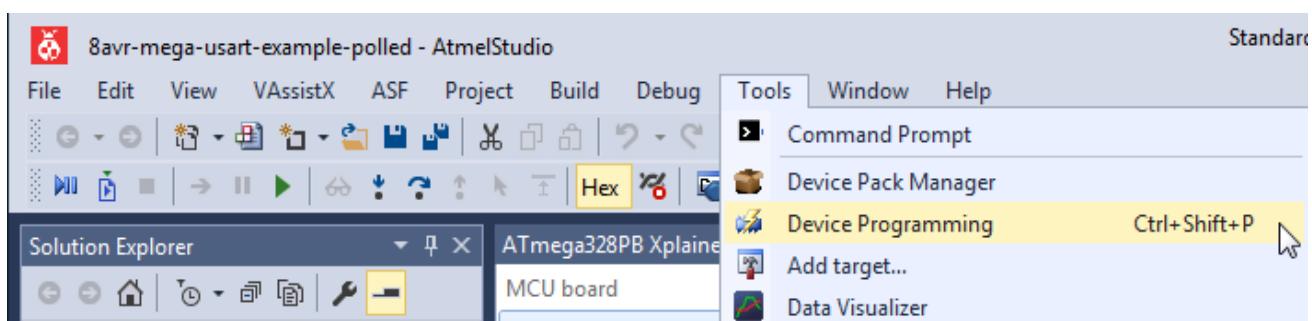


3 Programa los fusibles

Hay varias opciones clave de configuración de hardware que deben configurarse. Los siguientes ajustes de fusibles deben programarse en el dispositivo:

- ALTA: 0xDF
- BAJA: 0xC0
- EXT: 0xFC

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



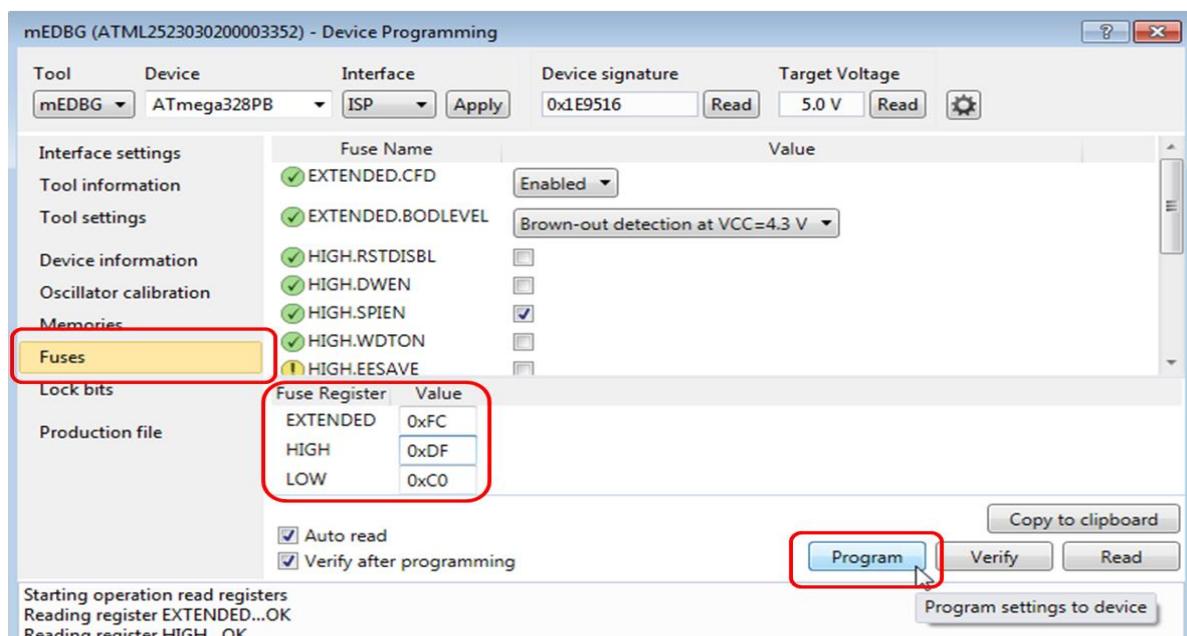
En el cuadro de diálogo Programación de dispositivos, seleccione la herramienta, el dispositivo y la interfaz como se muestra y, a continuación, pulse Aplicar:



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

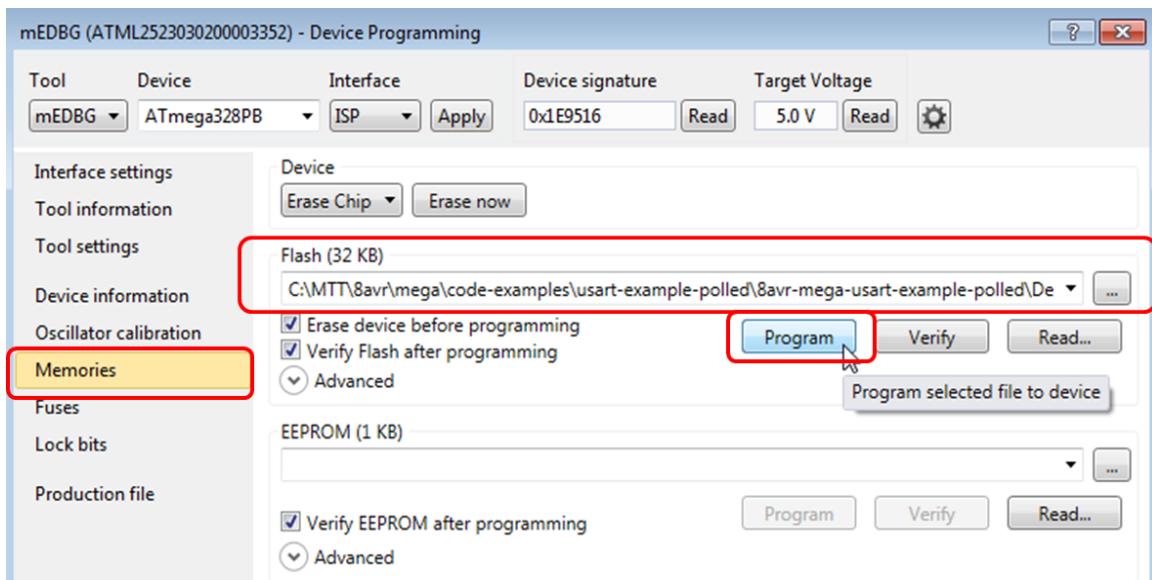


Seleccione la subsección Fusibles, Introduzca los 3 valores de bytes de fusibles anteriores y, a continuación, pulse Programa como se muestra:



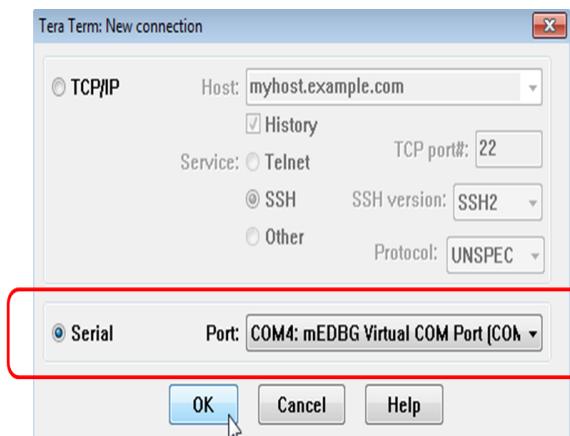
4 Programar el archivo hexadecimal

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. Programa de prensa como se muestra:

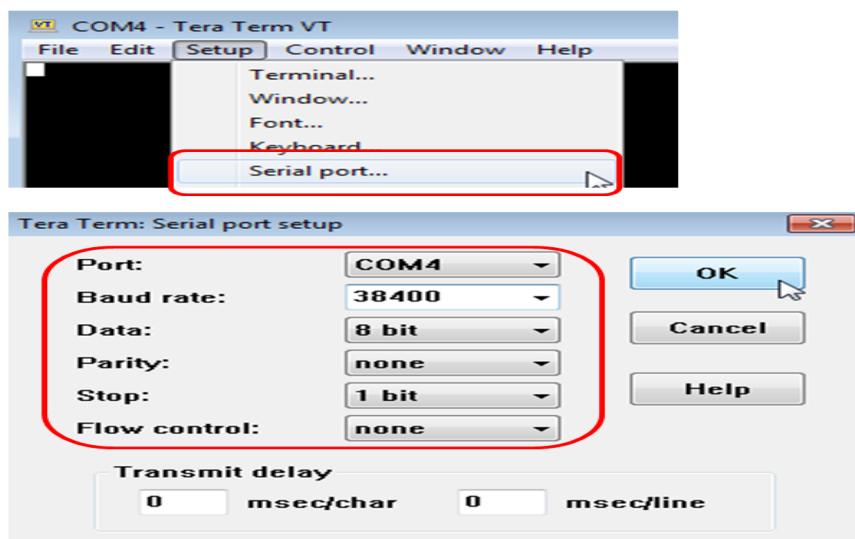


5

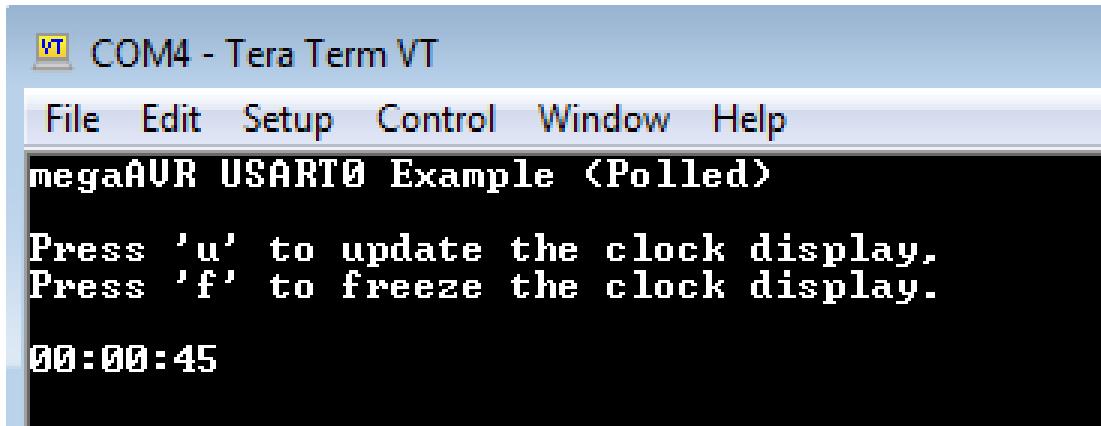
Inicie la aplicación Tera Term. Seleccione el puerto COM mEDBG correcto en el cuadro de diálogo que aparece:



A continuación, configure los parámetros del puerto serie para que coincidan con los valores predeterminados del proyecto: 38400 baudios, 8 datos, sin paridad, 1 parada, sin control de flujo como se muestra:



Resultados



VT COM4 - Tera Term VT

File Edit Setup Control Window Help

megaAVR USART0 Example <Polled>

Press 'u' to update the clock display.
Press 'f' to freeze the clock display.

00:00:45

The screenshot shows a terminal window titled "VT COM4 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays the title "megaAVR USART0 Example <Polled>". Below the title, instructions are given: "Press 'u' to update the clock display." and "Press 'f' to freeze the clock display.". At the bottom of the text area, the time "00:00:45" is displayed.

Es posible que vea galimatías en la pantalla después de restablecer los parámetros de comunicación. Simplemente realice un restablecimiento de la placa acortando RST a GND o reprogramando el archivo hexadecimal en la placa nuevamente (consulte el paso 4 anterior).

Conclusiones

Este proyecto ha proporcionado un ejemplo de cómo configurar y utilizar el módulo USART en el MCU megaAVR®.

Interrupciones

Descripción general de las 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 pines externos
 - Asociado a los pines de interrupción externos INT0-INT7
- Interrupciones de cambio de pin
 - Asociado con interrupciones externas PCINT0-PCINT2 que se producen en un cambio de pin de puerto

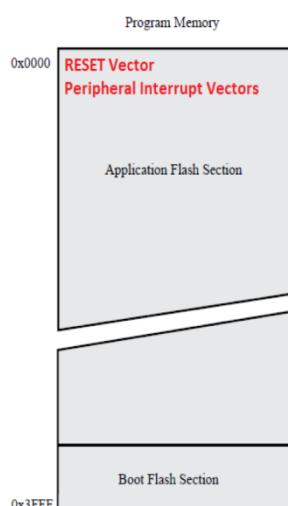
A los periféricos se les asignan bits individuales de habilitación de interrupciones en su respectivo registro de máscara de interrupciones que debe escribirse como lógico junto con el I-bit de habilitación de interrupción global en el Registro de estado para habilitar la interrupción.

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							

Restablecer e interrumpir ubicaciones vectoriales

Las fuentes reset & interrupt tienen cada una un vector de programa separado en el espacio de memoria del programa. Las direcciones más bajas en el espacio de memoria del programa se definen de forma predeterminada como los vectores de restablecimiento e interrupción como se muestra:



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 memoria del programa programando el bit de fusible BOOTRST en "0" y estableciendo el bit IVSEL del Registro de configuración del microcontrolador (MCUCR) en "1". La posible ubicación del vector RESET e interrupción se muestra aquí:

BOOTRST IVSEL Restablecer Addr. Interruptir Vector Start Addr.

1 0 0x0000 0x0002

1 1 0x0000 Boot Reset Addr. + 0x0002

0 0 Boot Reset Addr. 0x0002

0 1 Boot Reset Addr. Boot Reset Addr. + 0x0002

La dirección de restablecimiento 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

Los fusibles se programan utilizando un procedimiento de programación especial dentro de Atmel Studio 7 u otro programador.

Para evitar cambios involuntarios en las tablas de vectores de interrupción, se debe seguir un procedimiento de escritura especial para cambiar el bit IVSEL:

- Escriba el bit Interrupt Vector Change Enable (IVCE) en 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 void move_interrupts(void)
2{
3  uchar temp;
4  /* GET MCUCR */
5  temp = MCUCR;
6  /* Enable change of Interrupt Vectors */
7  MCUCR = temp | (1 << IVCE);
8  /* Move interrupts to Boot Flash section */
9  MCUCR = temp | (1 << IVSEL);
10 }
```

Nivel de prioridad

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

Procesamiento de interrupciones

Cuando se produce una interrupción, el bit I de habilitación de interrupción global se borra y todas las interrupciones se deshabilitan. El I-bit se establece automáticamente cuando se ejecuta una instrucción return from interrupt (RETI).

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

Básicamente hay 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 indicadores de interrupción.

Ejemplo USART Receive Complete Interrupt:

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 recepción completa de interrupciones (RXCIE) en UCSRnB, la interrupción usart de recepción completa se ejecutará siempre que se establezca el indicador RXC (siempre que las interrupciones globales estén habilitadas). Cuando se utiliza la recepción de datos controlada por interrupciones, la rutina completa de recepción 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.

Interrupciones no persistentes

Este tipo de interrupción se desencadena mediante 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 una lógica en las posiciones de los bits de la bandera que se van a borrar. Si se produce una condición de interrupción mientras se borra el bit de habilitación de interrupción correspondiente, el indicador de interrupción se establecerá y recordará hasta que se habilite la interrupción o el software borre el indicador. Del mismo modo, si se producen 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, a continuación, 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 hardware al ejecutar el vector de manejo de interrupciones correspondiente. Alternativamente, TOV0 se borra escribiendo uno lógico en la bandera. Cuando se establecen el SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable) y TOV0, se ejecuta la interrupción de desbordamiento de Timer/Counter0.

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 interrupciones de la biblioteca AVRLIBC.

Paso 1. #include Encabezados estándar

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

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

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 (I-bit en el registro de estado), así como una macro para asignar una función de interrupción a un vector de interrupción específico:

- `sei()`
- `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 controlador de interrupciones es diferente a una función ordinaria en el sentido de que maneja el contexto guardado y restaurado para garantizar que al regresar de la interrupción, se mantenga el contexto del programa. También se utiliza 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 le debe decir al compilador que use una forma alternativa de instrucción de retorno (`RETI` vs. `RET`)
- El compilador debe ser informado sobre cualquier opción adicional específica
 - Habilitar el anidamiento de interrupciones
 - Opciones para la generación de código de prólogo/epílogo
- La función debe estar vinculada 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 controlador de interrupciones con atributos

Para todos los vectores de interrupción sin controladores específicos, se instalará un controlador de interrupciones predeterminado: el controlador de interrupciones predeterminado restablecerá el dispositivo.

Una aplicación puede invalidar el controlador predeterminado y proporcionar un controlador de interrupción predeterminado específico de la aplicación mediante el BADISR_vect vector_id dentro de la macro ISR().

ISR() Macro

En el ejemplo de código siguiente se muestra cómo utilizar la macro ISR() para definir una función de interrupción:

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

Los diversos parámetros ahora se describirán más detalladamente.

vector_id

Este identificador es una *concatenación* de un ID de origen vectorial y _vect. Los ID de origen vectorial 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

Misspelt vector_ids seguirá generando una función, sin embargo, no se conectará a la tabla de vectores de interrupción. El compilador generará una advertencia si detecta un nombre de aspecto sospechoso.

Atributos

Los atributos ISR() proporcionan instrucciones adicionales 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 ningún atributo especificado

ISR_NOBLOCK

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

Esto se puede utilizar para crear ISR anidados, sin embargo, se debe tener cuidado para evitar desbordamientos de pila, o para evitar entrar infinitamente en el ISR para aquellos casos en que el hardware AVR no borra la marca de interrupción respectiva antes de ingresar al ISR.

ISR_NAKED

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

ISR_ALIASOF(vector_id)

Esto se puede utilizar para definir vectores adicionales que comparten el mismo controlador. En el ejemplo siguiente se alias el vector PCINT1 al controlador PCINT0:

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

Ejemplo isr()

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

```

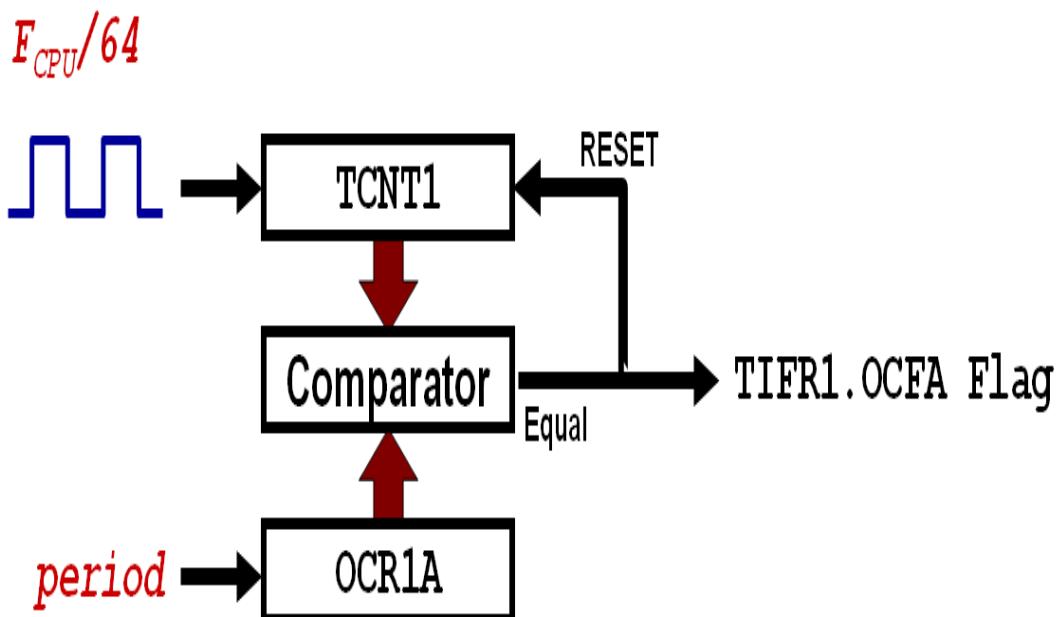
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 ISR(TIMER1_COMPA_vect, ISR_BLOCK)
5 {
6     PORTB ^= (1 << PORTB5); // Toggle LED0
7 }
8
9 int main(void)
10{
11    // Initialization
12
13    // Set LED as output
14    DDRB |= (1 << PORTB5); // Configure PB5 as digital output
15    PORTB &= ~(1 << PORTB5); // Set initial level for PB5
16
17    // Set up Timer/Counter1
18    TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
19    OCR1A = 25000; // Set CTC compare value to 10Hz (100mS)
20                                // at 16MHz AVR clock, with a prescaler of 64
21    TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt
22    TCCR1B |= ((1 << CS10) | (1 << CS11)); // Start Timer/Counter1 at F_CPU/64
23
24    // Enable all interrupts
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 Timer/Counter. Cada módulo tiene un modo llamado Clear Timer on Compare (CTC) que, cuando se inicializa correctamente, activará periódicamente una señal de marca de coincidencia de comparación de salida de temporizador1 en el registro de indicador de interrupción TIFR1 (TIFR1.OCF1A Flag) como se muestra:



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

```
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 ISR(TIMER1_COMPA_vect, ISR_BLOCK)
5 {
6     PORTB ^= (1 << PORTB5); // Toggle LED0
7 }
8
9 int main(void)
10 {
11     // Initialization
12
13     // Set LED as output
14     DDRB |= (1 << PORTB5); // Configure PB5 as digital output
15     PORTB &= ~(1 << PORTB5); // Set initial level for PB5
16
17     // Set up Timer/Counter1
18     TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
19     OCR1A = 25000; // Set CTC compare value to 10Hz (100mS)
20                     // at 16MHz AVR clock, with a prescaler of 64
21     TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt
22     TCCR1B |= ((1 << CS10) | (1 << CS11)); // Start Timer/Counter1 at F_CPU/64
23
24     // Enable all interrupts
25     sei();
26
27     while(1);
28 }
```

Este es un ejemplo de una interrupción no persistente. El TIFR1. El indicador OCFA es borrado automáticamente por el hardware al ingresar al controlador.

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

Paso 4. Habilitar todas las interrupciones

Finalmente, necesitamos habilitar globalmente todas las interrupciones periféricas habilitadas configurando el I-bit 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 interrupciones globalmente

```
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 ISR(TIMER1_COMPA_vect, ISR_BLOCK)
5 {
6     PORTB ^= (1 << PORTB5); // Toggle LED0
7 }
```

```
8 int main(void)
9 {
10    // Initialization
11
12    // Set LED as output
13    DDRB |= (1 << PORTB5);      // Configure PB5 as digital output
14    PORTB &= ~(1 << PORTB5);    // Set initial level for PB5
15
16    // Set up Timer/Counter1
17    TCCR1B |= (1 << WGM12 );    // Configure timer 1 for CTC mode
18    OCR1A = 25000;              // Set CTC compare value to 10Hz (100mS)
19                                // at 16MHz AVR clock, with a prescaler of 64
20    TIMSK1 |= (1 << OCIE1A );  // Enable CTC interrupt
21    TCCR1B |= ((1 << CS10 ) | (1 << CS11 )); // Start Timer/Counter1 at F_CPU/64
22
23    // Enable all interrupts
24    sei();
25
26    while(1);
27 }
```

Ejemplo de código de interrupción megaAVR®

Objetivo

Esta página proporciona un ejemplo de código de interrupción básico para la MCU ATmega328PB. El proyecto configura el módulo Timer/Counter1 para que funcione 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 "tick" que es utilizada por el bucle principal para alternar LED0 cada 100 mS.

Materiales

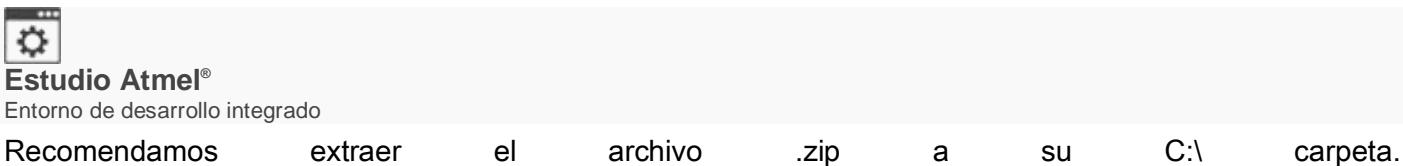
Herramientas de hardware



ATmega328PB Xplained Mini

Kit de evaluación

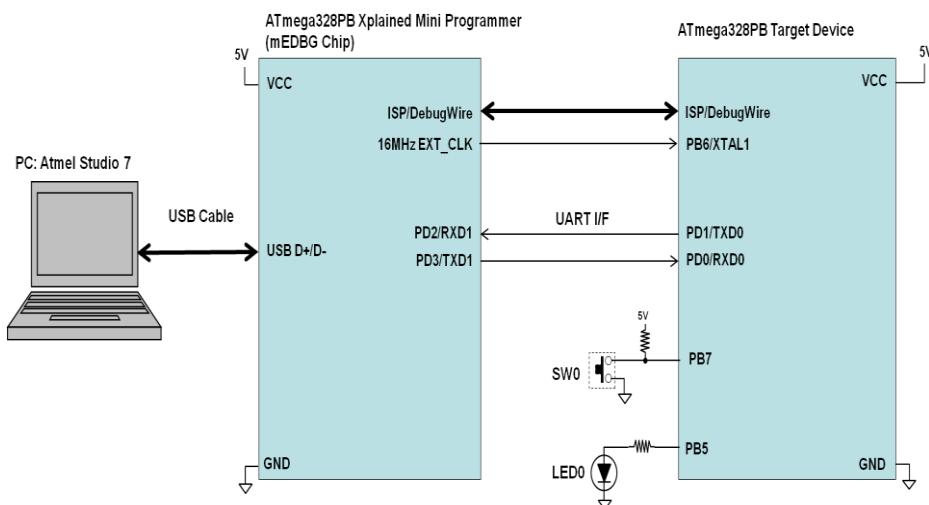
Herramientas de software



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

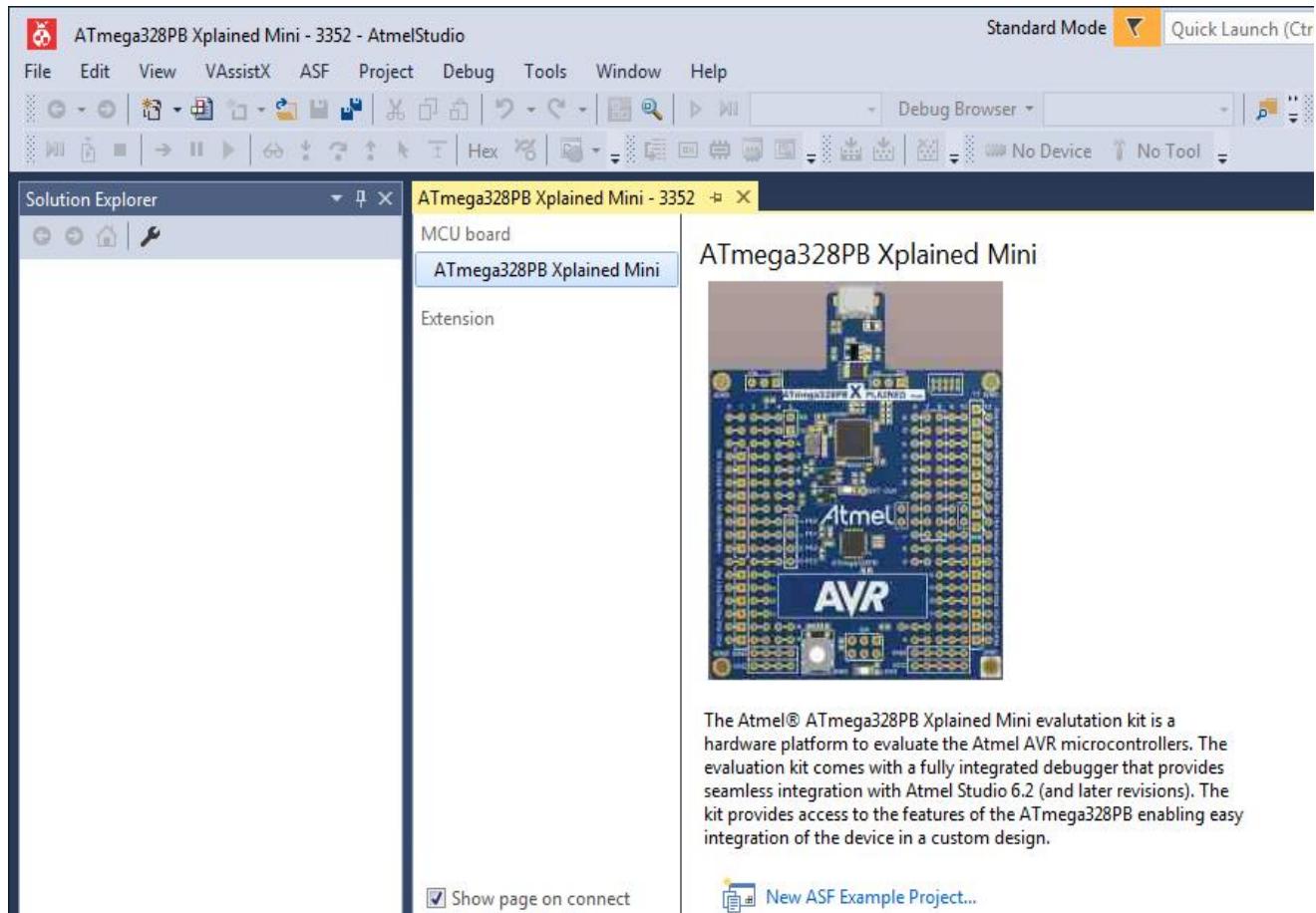
Diagrama de conexión

El módulo USART0 del dispositivo ATmega328PB de destino está conectado a la interfaz USART del chip mEDBG. El chip mEDBG realiza la conversión serie USB 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 suministrar un reloj de 16MHz cuando la placa Xplained se conecta a través de un cable USB a una PC. El LED0 está conectado al puerto PB5 como se muestra:



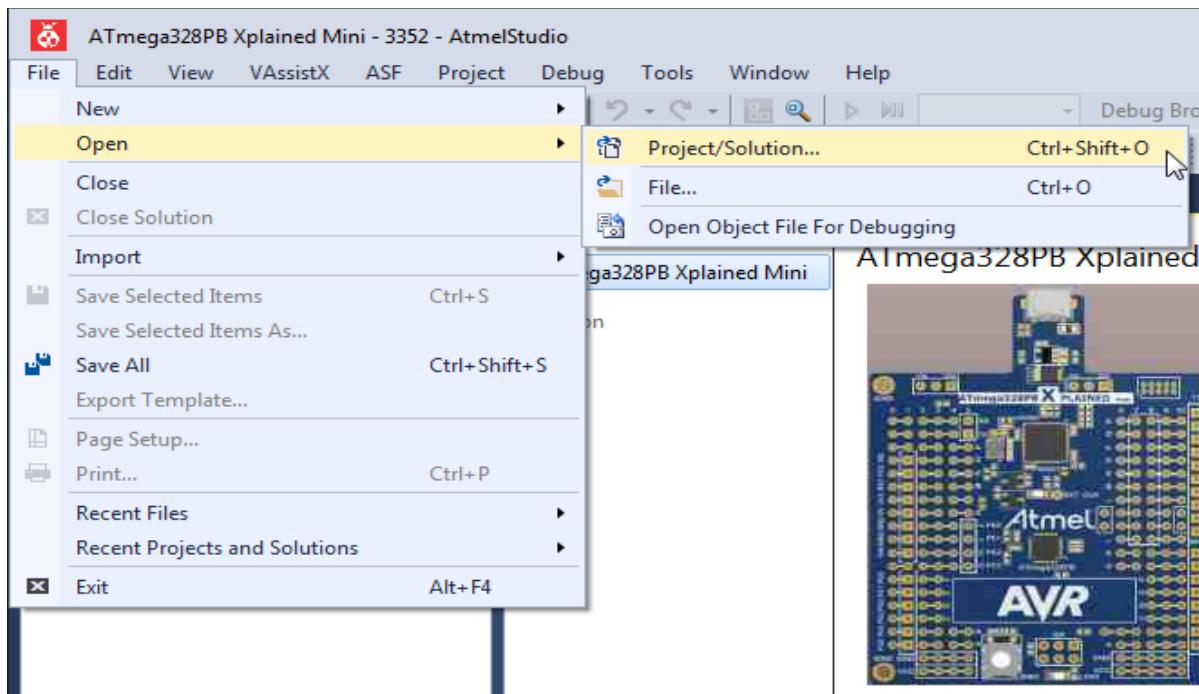
Procedimiento

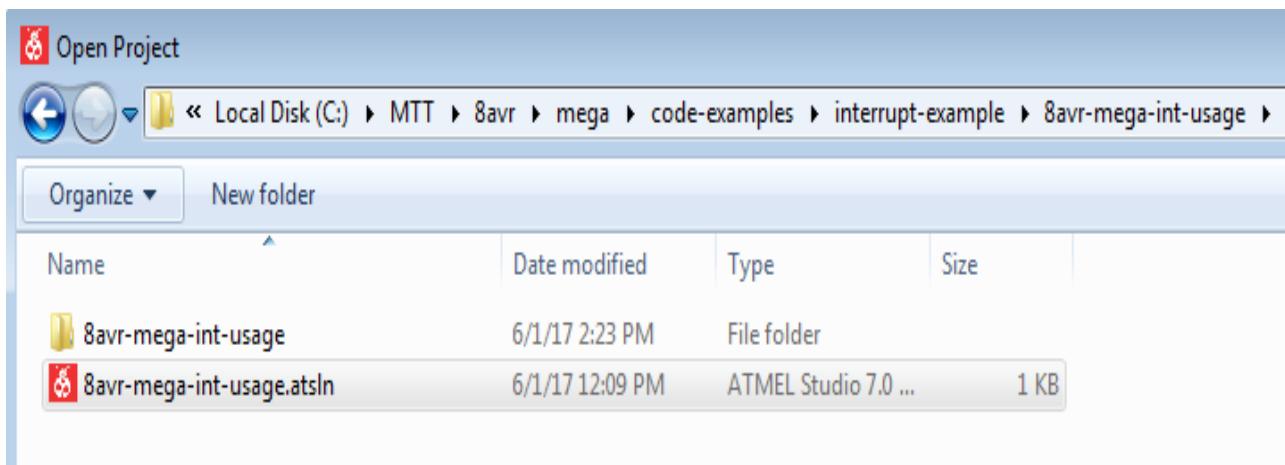
Conecte la placa ATmega328PB Xplained Mini a su computadora con un cable USB A a MicroB. Inicie Atmel Studio 7. Si la placa se ha enumerado correctamente, debería ver la imagen de la placa aparecer en Studio como se muestra:



El tablero 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"

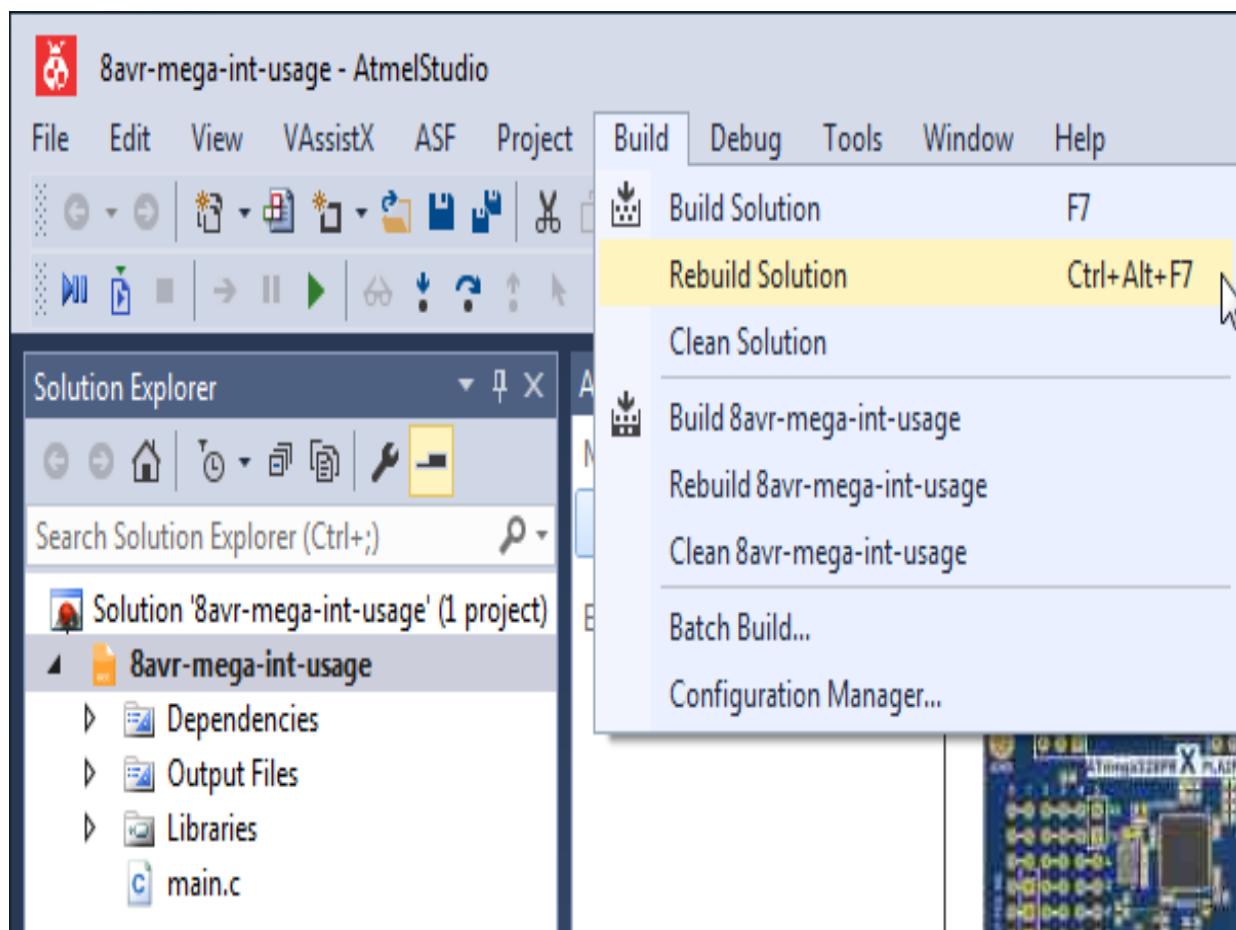
1 Abrir la solución





Para comprender cómo se configuraron y habilitaron las interrupciones en este ejemplo (archivo main.c), revise la página Configuración de interrupciones de megaAVR®.

2 Reconstruir la solución

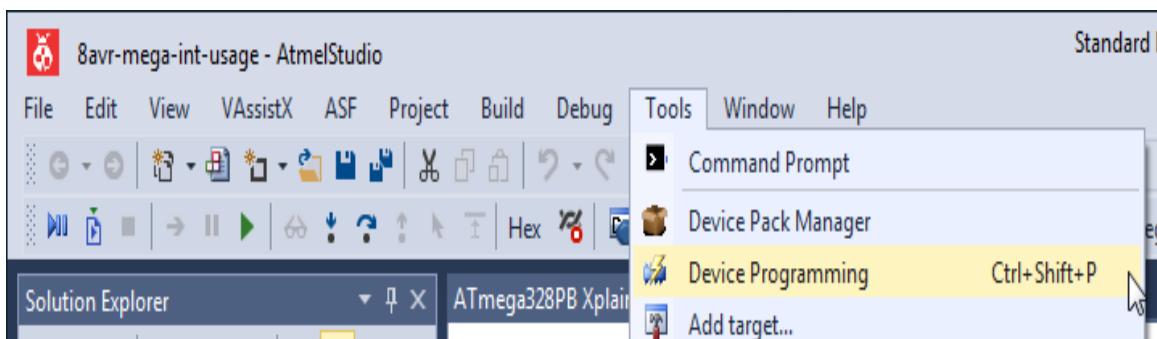


3 Programa los fusibles

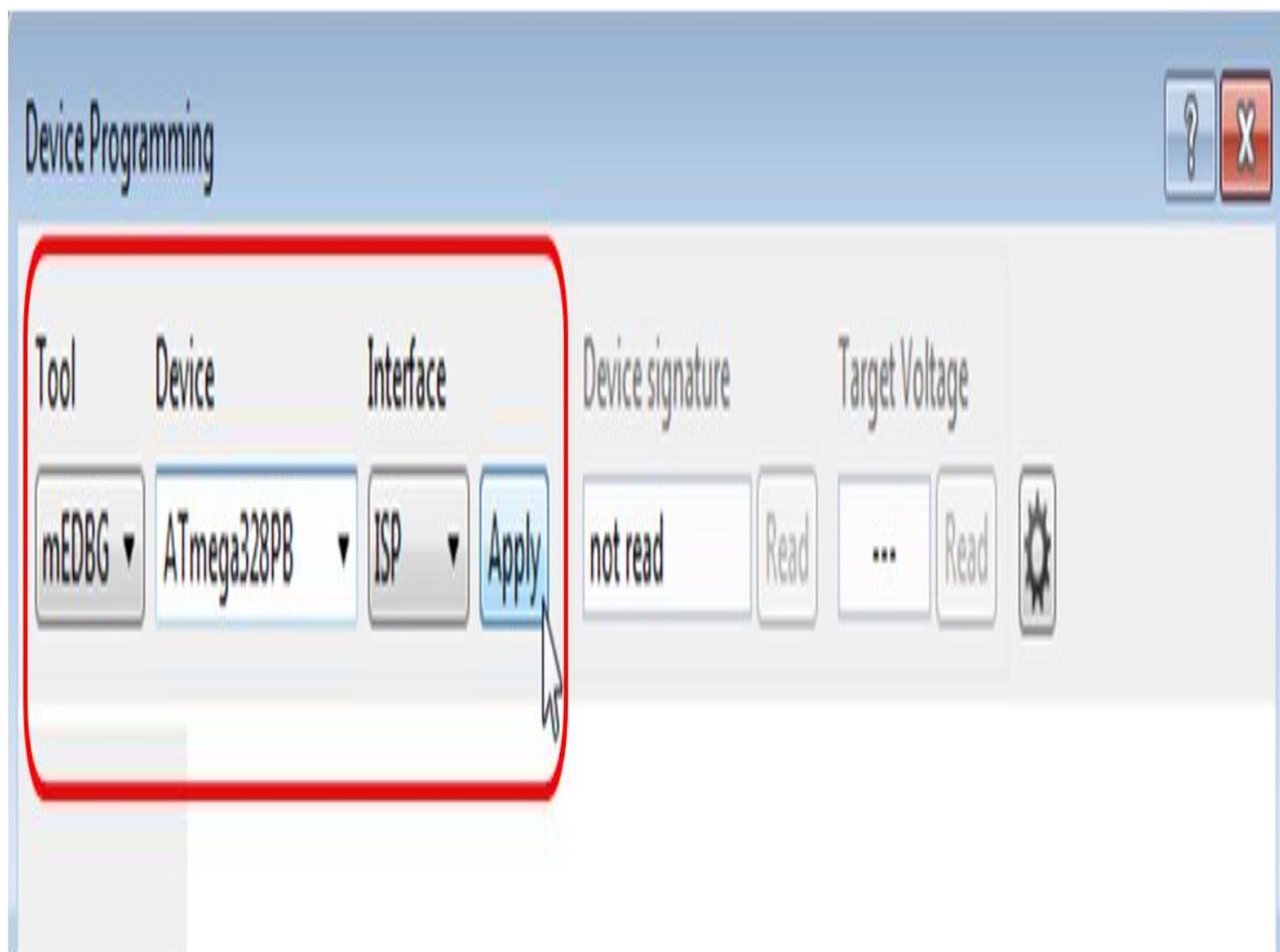
Hay varias opciones clave de configuración de hardware que deben configurarse. Los siguientes ajustes de fusibles deben programarse en el dispositivo:

- ALTA: 0xDF
- BAJA: 0xC0
- EXT: 0xFC

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



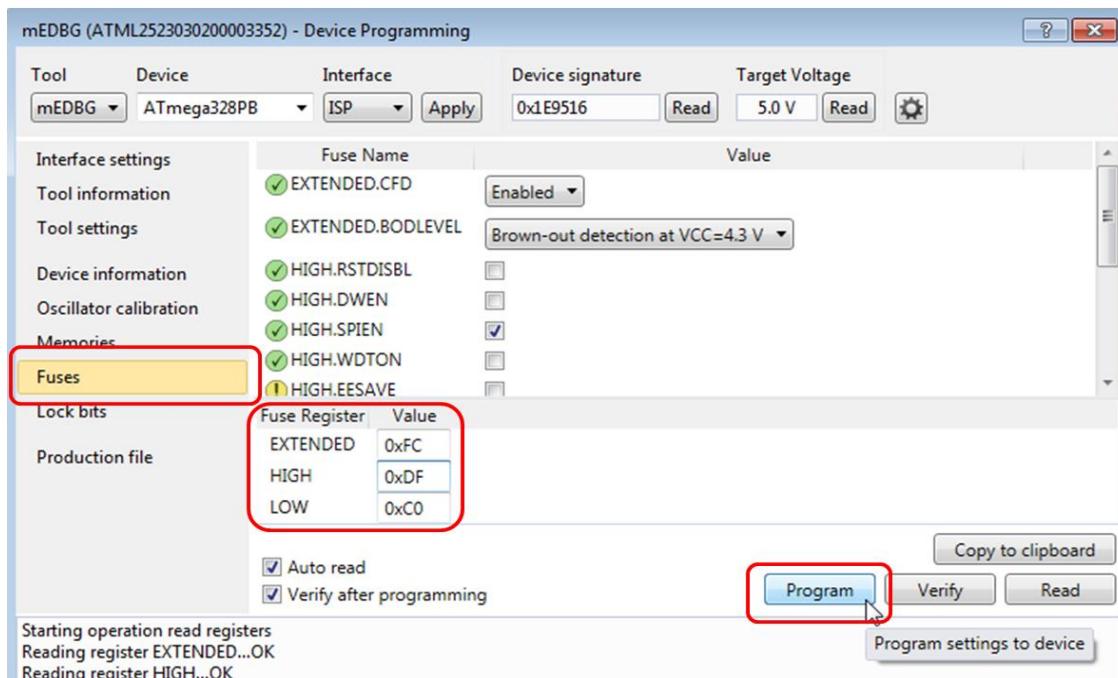
En el cuadro de diálogo Programación de dispositivos, seleccione la herramienta, el dispositivo y la interfaz como se muestra y, a continuación, pulse Aplicar:



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

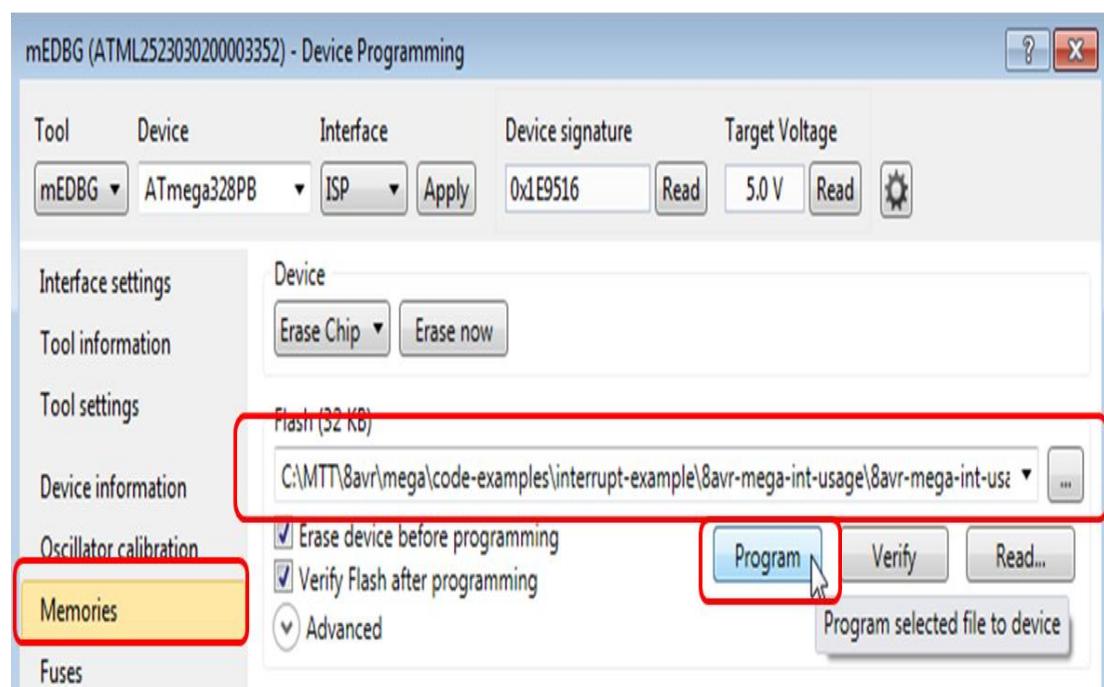


Seleccione la subsección Fusibles, Introduzca los 3 valores de bytes de fusibles anteriores y, a continuación, pulse Programa como se muestra:

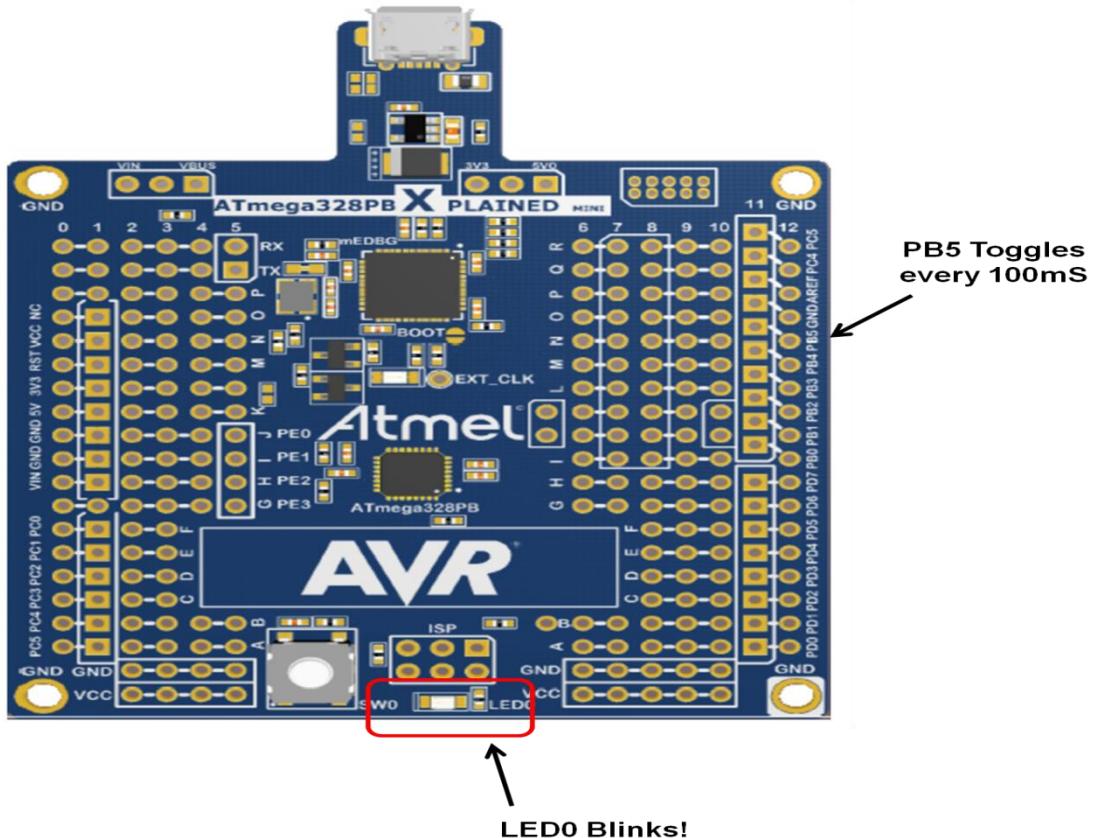


4 Programar el archivo hexadecimal

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



Resultados



Conclusiones

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

Consideraciones especiales

Esta página cubre algunas consideraciones especiales a tener en cuenta al trabajar con interrupciones en MCU AVR.

Compartir datos con el ISR

Las variables compartidas entre el ISR y el programa principal deben declararse 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 normalmente accederá a "flag" solo una vez, y optimizará los accesos adicionales por completo, ya que su análisis de la ruta de código muestra que nada dentro del bucle podría cambiar el valor de "flag" de todos modos.

Para indicar al compilador que esta variable podría cambiarse fuera del ámbito 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 mencionó anteriormente, el compilador se asegura de que dondequiero 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 se considere atómica, debe garantizar el acceso ininterrumpido de una variable dada. Muchos lenguajes ensambladores proporcionan esto en ciertos niveles, es decir, prueba y conjunto de bits, sin embargo, no hay ninguna disposición para proporcionar automáticamente la atomicidad de todos los tipos de variables en el lenguaje ANSI C.

Las expresiones y sentencias 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 dicha variable como volátil asegura 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.h>
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 bucle `while()` cuando la variable `ctr` simplemente alcance el valor `0x00FF`. Esto sucede porque el compilador no puede acceder de forma nativa a una variable de 16 bits atómicamente en una CPU de 8 bits. Entonces, cuando `ctr` está, por ejemplo, en `0x0100`, el compilador prueba el byte bajo para 0, lo que tiene éxito. Luego procede a probar el byte alto, pero en ese momento el ISR se activa y el contexto principal se interrumpe. El ISR disminuirá la variable de `0x0100` a `0x00FF`, luego procederá el contexto principal. Ahora prueba el alto byte de la variable que es (ahora) también 0, por lo que concluye que la variable ha alcanzado 0 y termina el bucle.

Macros de acceso atómico

La biblioteca atómica AVR-LIBC proporciona la `ATOMIC_BLOCK` macros que insertan la protección de interrupción adecuada cuando se desea acceso atómico. Estas macros funcionan mediante 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 al ingresar al 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    uint_16 ctr_copy;
16    ...
17    ctr = 0x0200;
18    start_timer();
19    do
20    {
21        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
22        {
23            ctr_copy = ctr;
24        }
25    } while(ctr != 0);
26    // wait
27    ;
28 ...
29 }
```

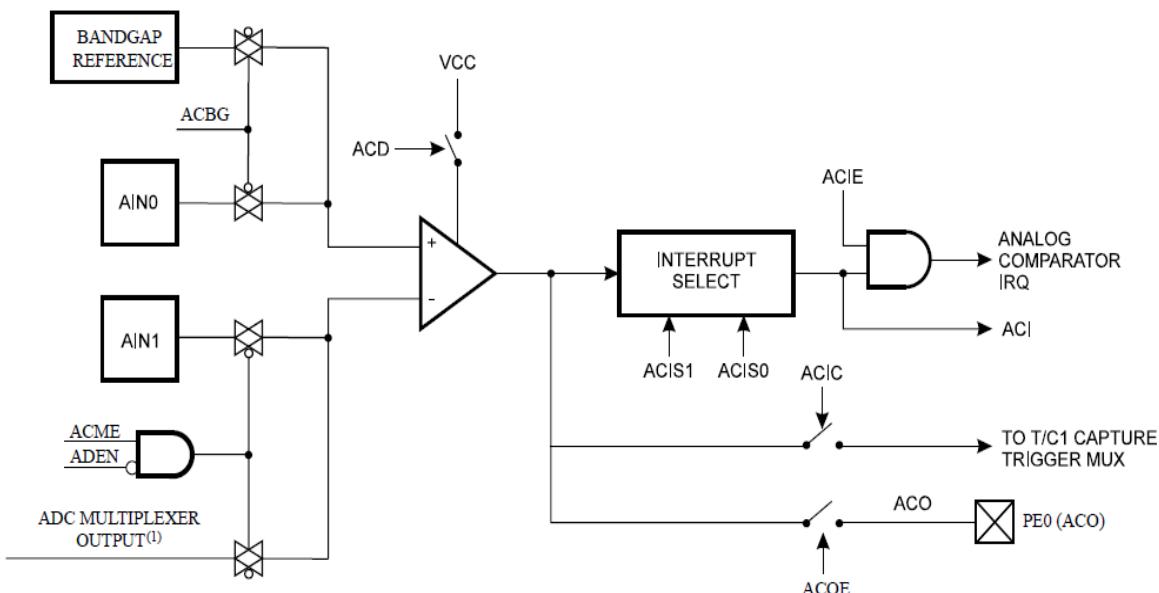
La macro `ATOMIC_BLOCK` instalará la protección contra interrupciones adecuada antes de acceder a la variable `ctr`, por lo que se garantiza que se probará de manera consistente. En este caso, el parámetro `ATOMIC_RESTORESTATE` hace que el `ATOMIC_BLOCK` restaure el estado anterior del registro SREG,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

Comparador analógico y referencia de voltaje

Descripción general de 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 Timer/Counter1. Además, el comparador puede activar una interrupción separada, exclusiva del comparador analógico. El usuario puede seleccionar La activación de interrupciones en la salida del comparador sube, baja o alterna.

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



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

Entrada multiplexada de 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 estar desactivado para utilizar esta función. Si el multiplexor de comparador analógico enable bit en el registro de control y estado B de ADC (ADCSR.B. 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 de multiplexores ADC (ADMUX. MUX[2..0]) seleccione el pin de entrada para reemplazar la entrada negativa al comparador analógico, como se muestra en la tabla siguiente. Cuando ADCSR.B. 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

Control de comparador analógico y registro de estado B

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

Nombre: ACSR B Offset: 0x4F Reset: 0x00.

Propiedad:

Al direccionar como registro de E/S: el desplazamiento de dirección es 0x2F



- Bit 0 – ACOE: Analog Comparator Output Enable

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

Control de comparador analógico y registro de estado

Al abordar los registros de E/S como espacio de datos utilizando instrucciones LD y ST, se debe utilizar el desplazamiento proporcionado. Cuando se utilizan los comandos específicos de E/S IN y OUT, el desplazamiento se reduce en 0x20, lo que da como resultado un desplazamiento de dirección de E/S dentro de 0x00 - 0x3F.

Nombre: ACSR Offset: 0x50 Restablecer: N/A

Propiedad:

Cuando se dirige como registro de E/S: el desplazamiento de dirección es 0x30

Bit	7	6	5	4	3	2	1	0
Access	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	a	0	0	0	0	0

- Bit 7 – ACD: Desactivación del comparador analógico

Cuando este bit se escribe como lógico, la alimentación del comparador analógico se apaga. Este bit se puede configurar en cualquier momento para desactivar el comparador analógico. Esto reducirá el consumo de energía en modo activo e inactivo. Al cambiar el bit ACD, la interrupción del comparador analógico debe deshabilitarse borrando el bit ACIE en ACSR. De lo contrario, puede producirse una interrupción cuando se cambia el bit.

- Bit 6 – ACBG: Comparador analógico Bandgap Select

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

- Bit 5 – ACO: Salida de comparador analógico

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

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

Este bit se establece por hardware cuando un evento de salida del comparador desencadena 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 al ejecutar el vector de manejo de interrupciones correspondiente. Alternativamente, ACI se borra escribiendo uno lógico en la bandera.

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

Cuando se escribe el bit ACIE logic one y se establece el I-bit en el Status Register, se activa la interrupción del comparador analógico. Cuando se escribe lógica cero, la interrupción se deshabilita.

- Bit 2 – ACIC: Analog Comparator Input Capture Enable

Cuando se escribe como lógica, este bit permite que la función de captura de entrada en Timer/Counter1 sea activada por el Comparador analógico. En este caso, la salida del comparador está conectada directamente a la lógica front-end de captura de entrada, lo que hace que el comparador utilice el cancelador de ruido y las funciones de selección de borde de la interrupción de captura de entrada

Timer/Counter1. Cuando se escribe lógica cero, no existe ninguna conexión entre el comparador analógico y la función de captura de entrada. Para que el comparador active la interrupción de captura de entrada del temporizador/contador1, se debe establecer el bit ICIE1 en el registro de máscara de interrupción del temporizador (TIMSK1).

- Bits 1:0 – ACISn: Analog Comparator Interrupt Mode Select [n = 1:0]

Estos bits determinan qué eventos de comparación desencadenan 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.

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

Registro de desactivación de entrada digital 1

Nombre: DIDR1 Offset: 0x7F; Restablecer: 0x00.

Propiedad:

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

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

Cuando este bit se escribe lógica uno, 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 lógicamente uno para reducir el consumo de energía en el búfer de entrada digital.

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

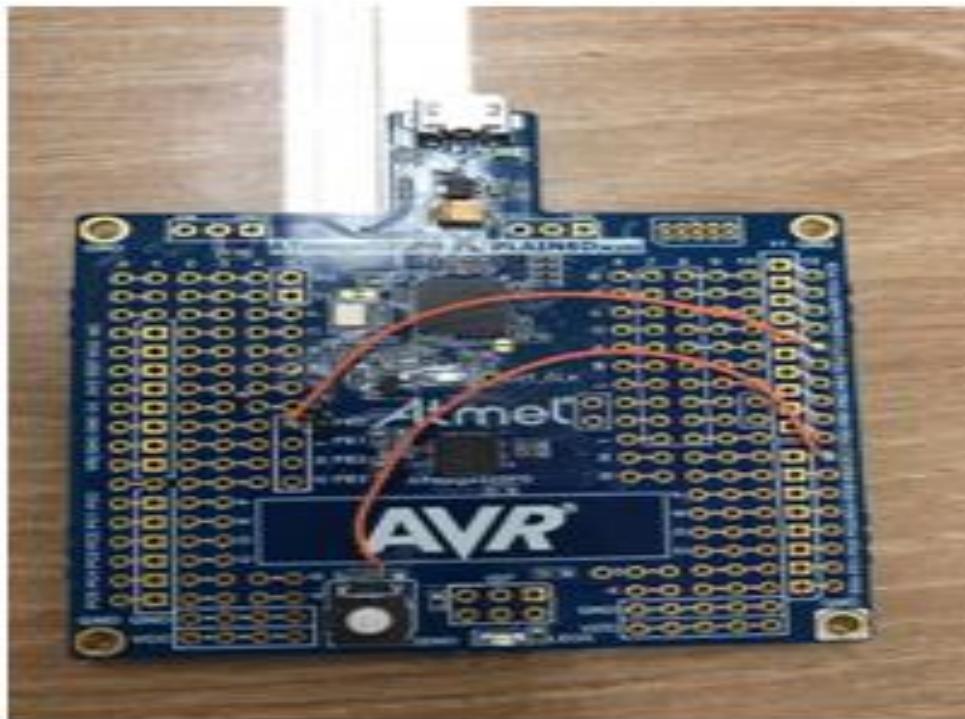
Symbol	Parameter	Condition	Min.	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)		1.1	1.5	1.7	V
	Power-on Reset Threshold Voltage (falling) ⁽²⁾		0.6	1.0	1.7	V
SR_{ON}	Power-on Slope Rate		0.01	-	10	V/ms
V_{RST}	RESET Pin Threshold Voltage		0.2 V_{CC}	-	0.9 V_{CC}	V
t_{RST}	Minimum pulse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

Comparador analógico y ejemplo de referencia de voltaje

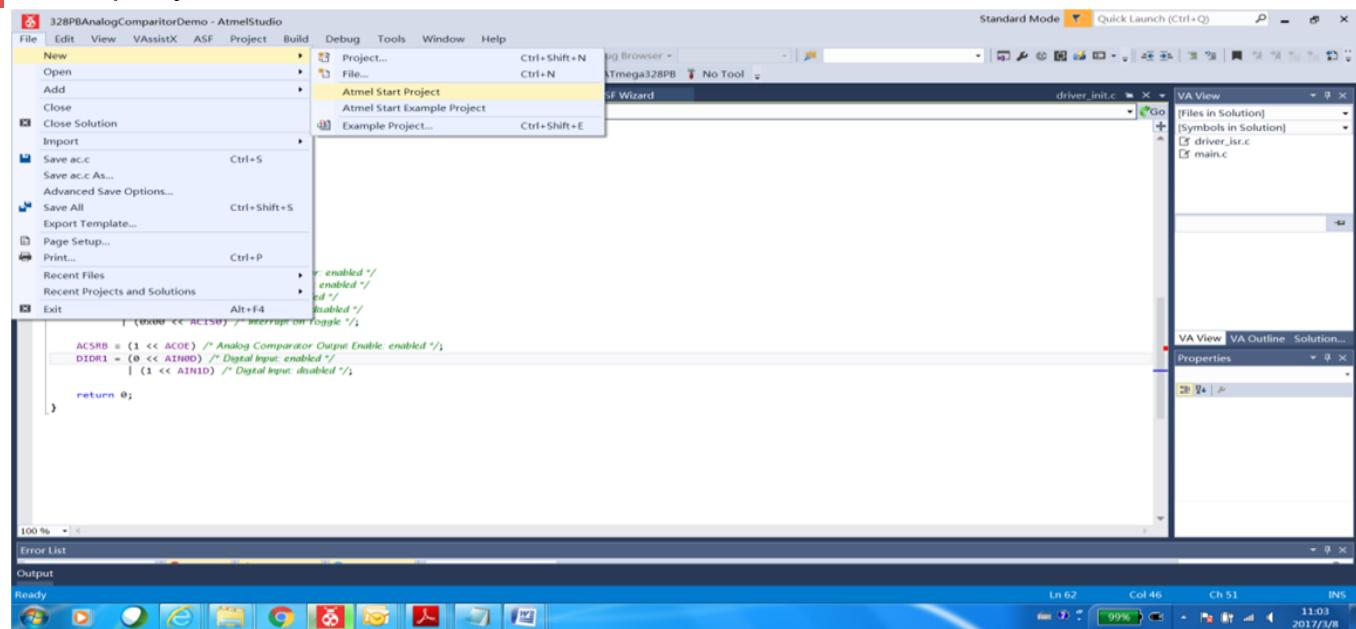
Este proyecto práctico demuestra un ejemplo simple de comparador analógico y voltaje de referencia: referencia de intervalo de banda como entrada positiva (1.1V) y AIN1 como entrada negativa. Al cablear AIN1 (PD7) al conmutador (SW0, PB7), que se enciende a alto (5 V) y se apaga a GND, y cablear ACO (PE0) a PB5 (LED0), se puede observar la salida del comparador (ACO, PE0) verificando el estado del LED.

- Cableado: PD7 (AIN1) | PB7 (SW0; | PE0 (ACO) PB5 (LED0)

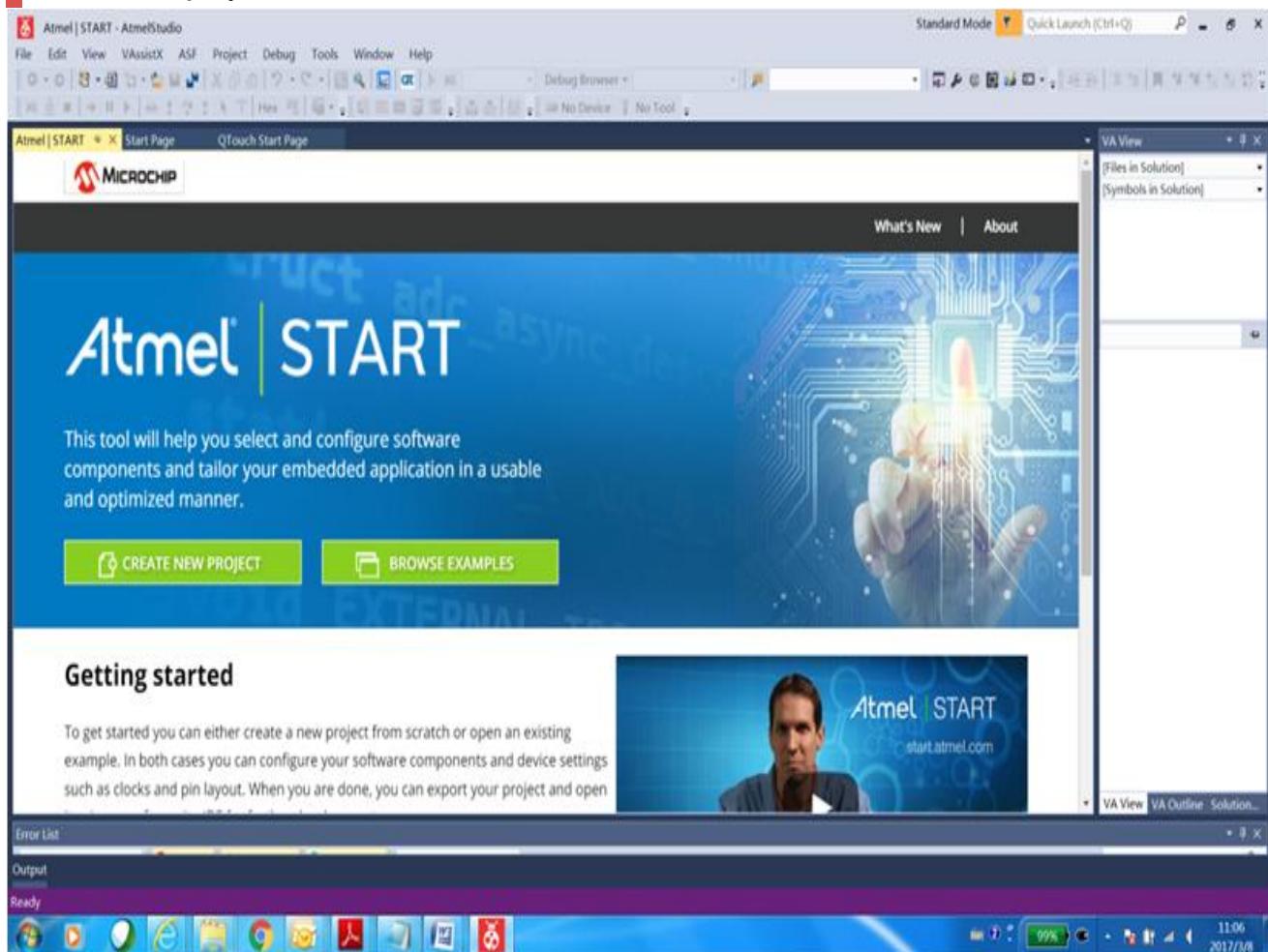
Tablero Xplained:



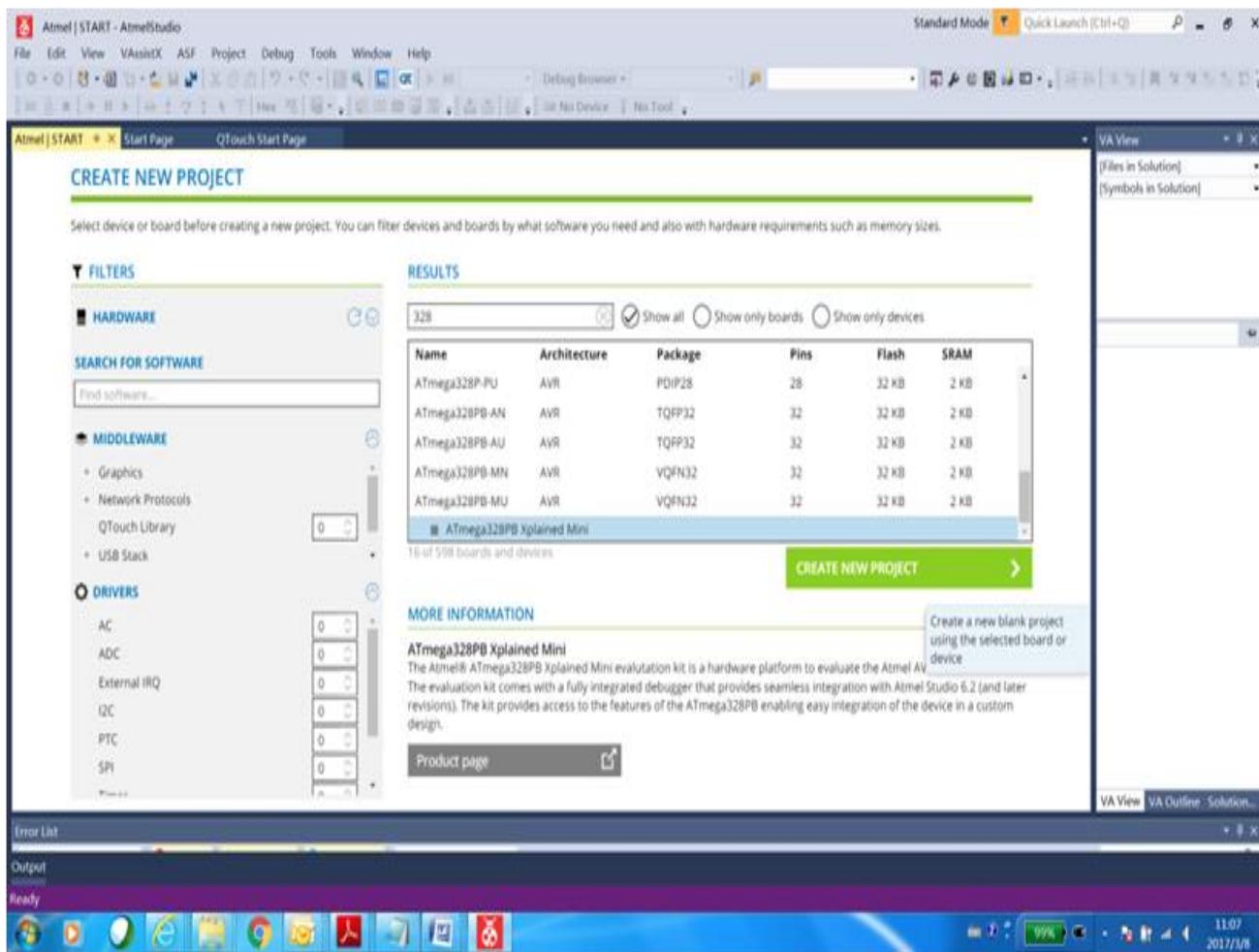
1 Nuevas | Proyecto Atmel Start



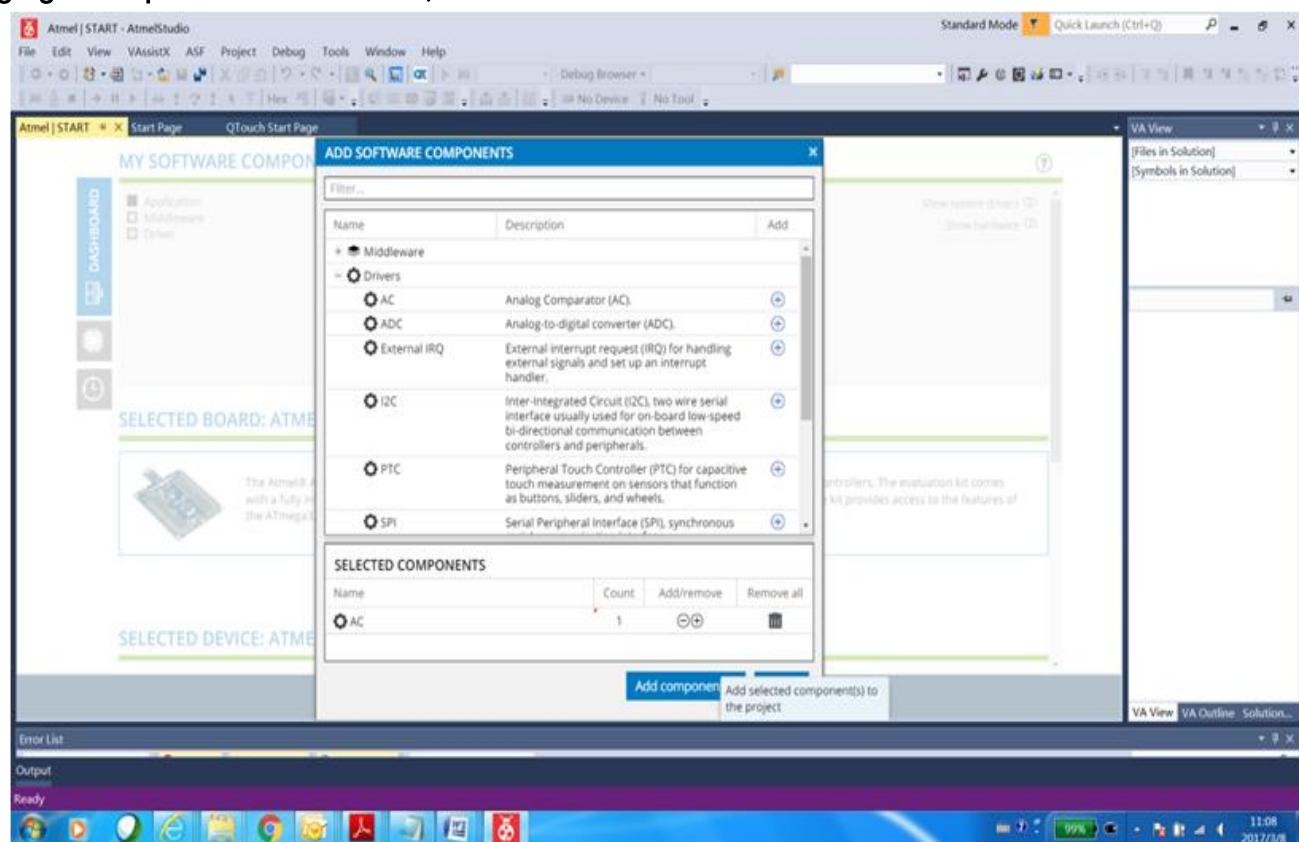
2 Crear nuevo proyecto



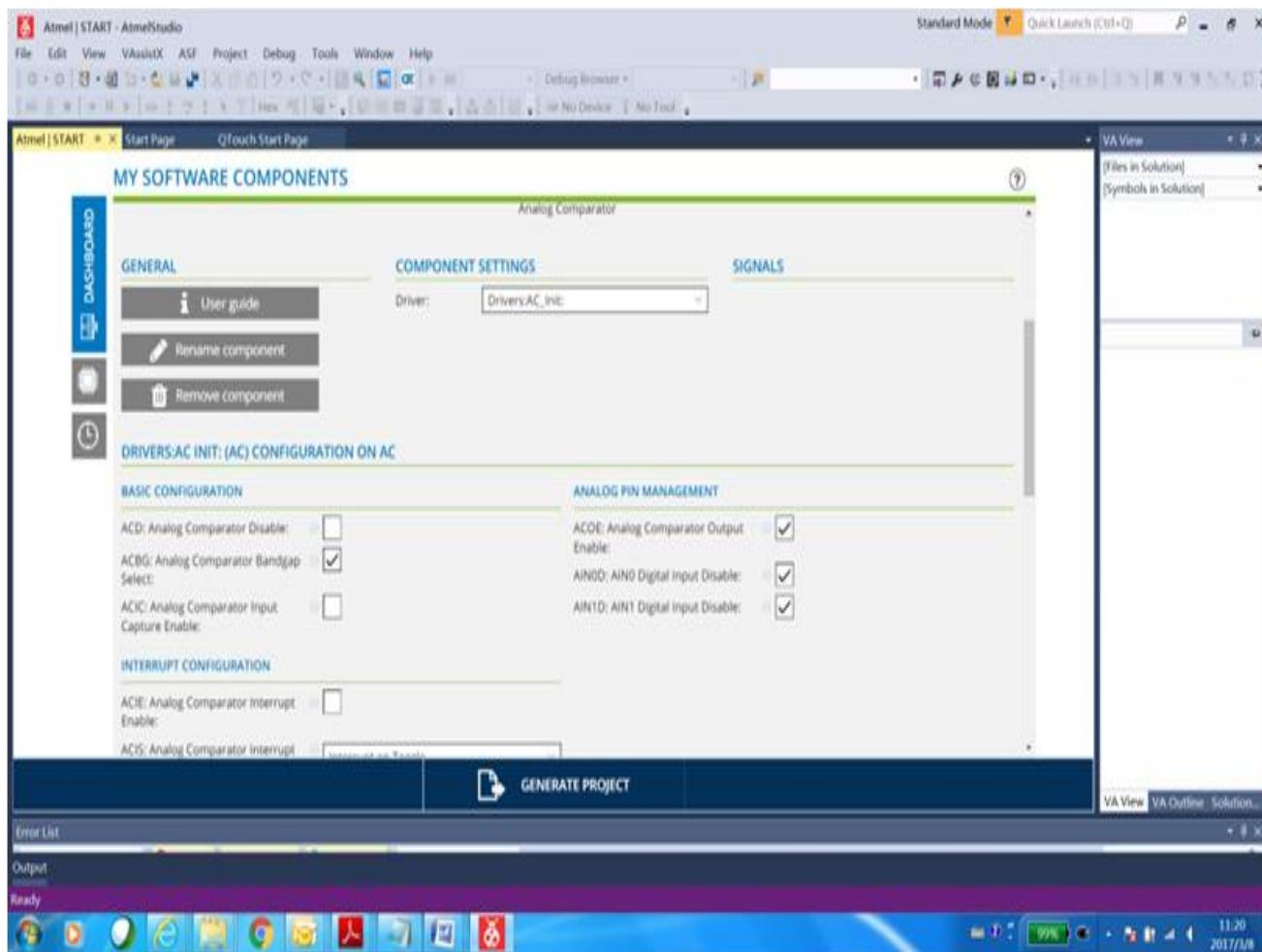
3 Seleccione ATmega328PB Xplained Mini:



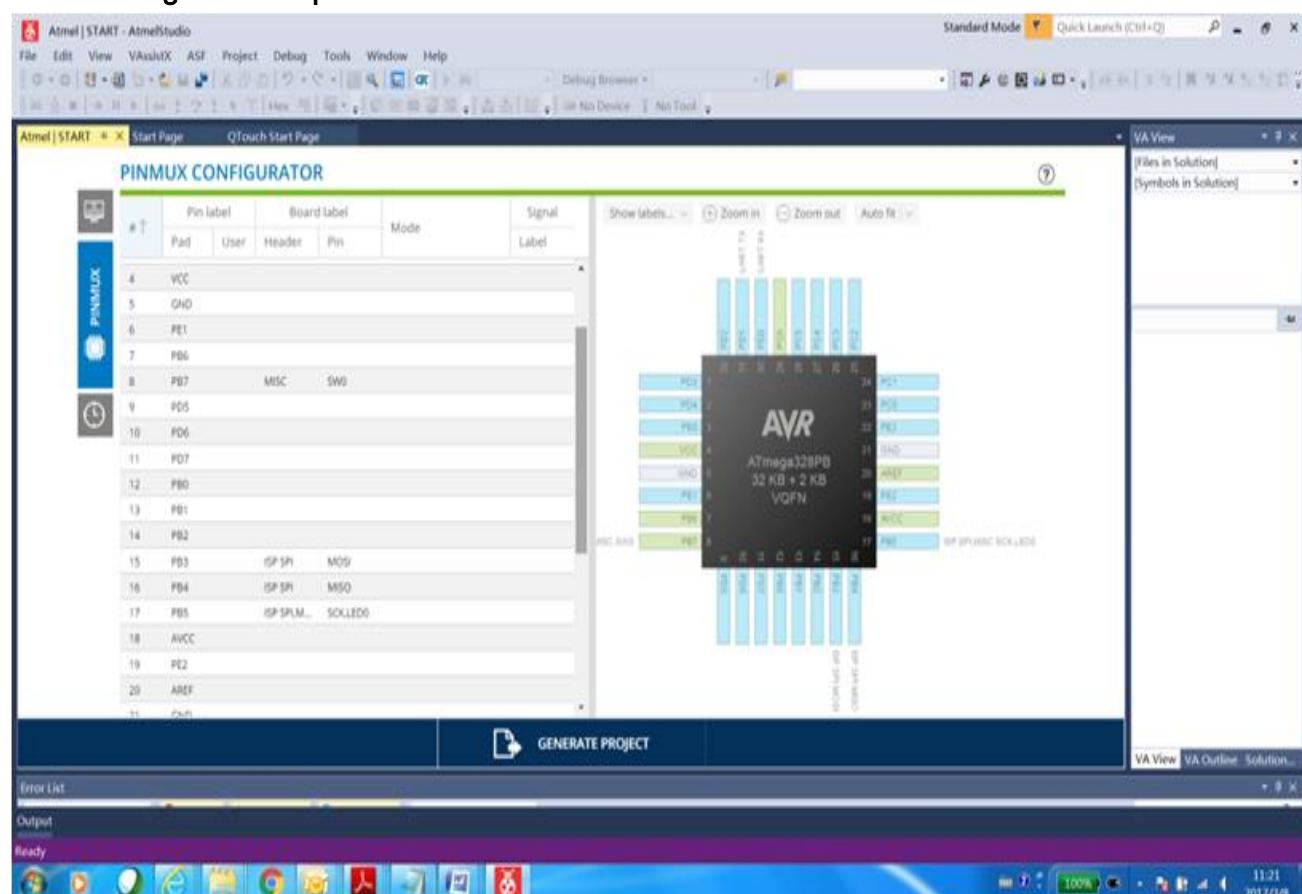
4 Agregar componentes de software, seleccione CA:



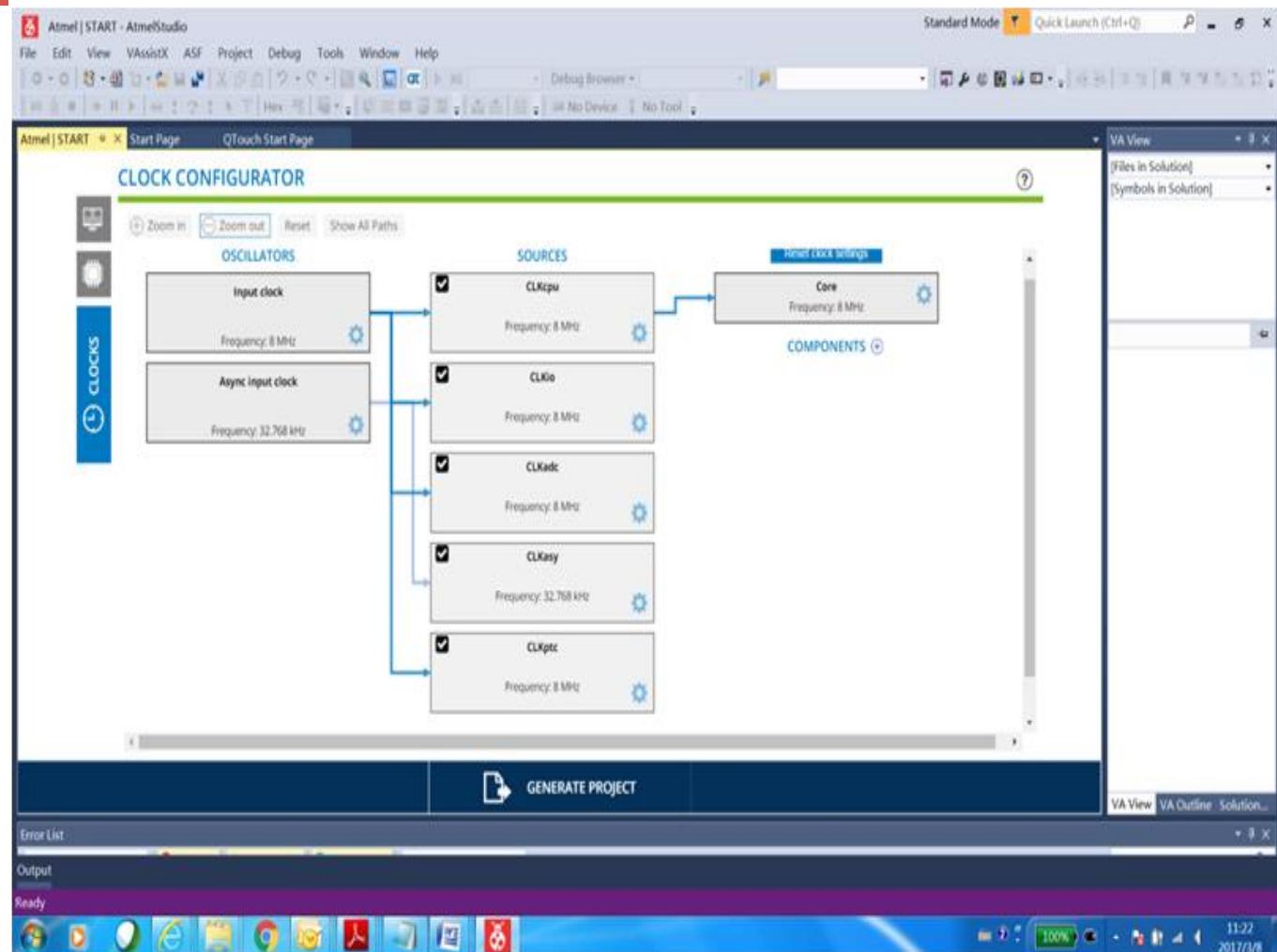
5Configuración de CA:



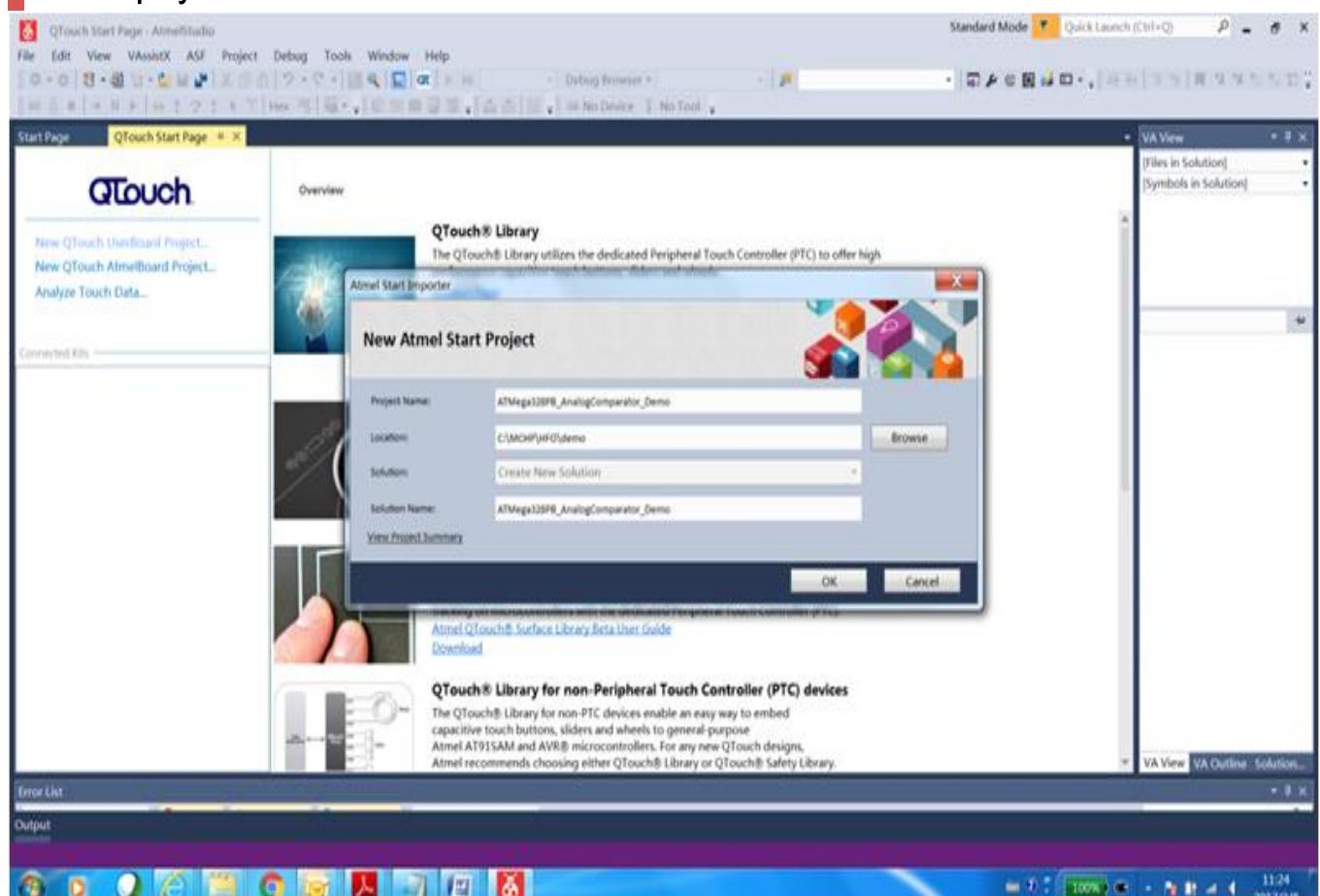
6Vista del configurador de pines:



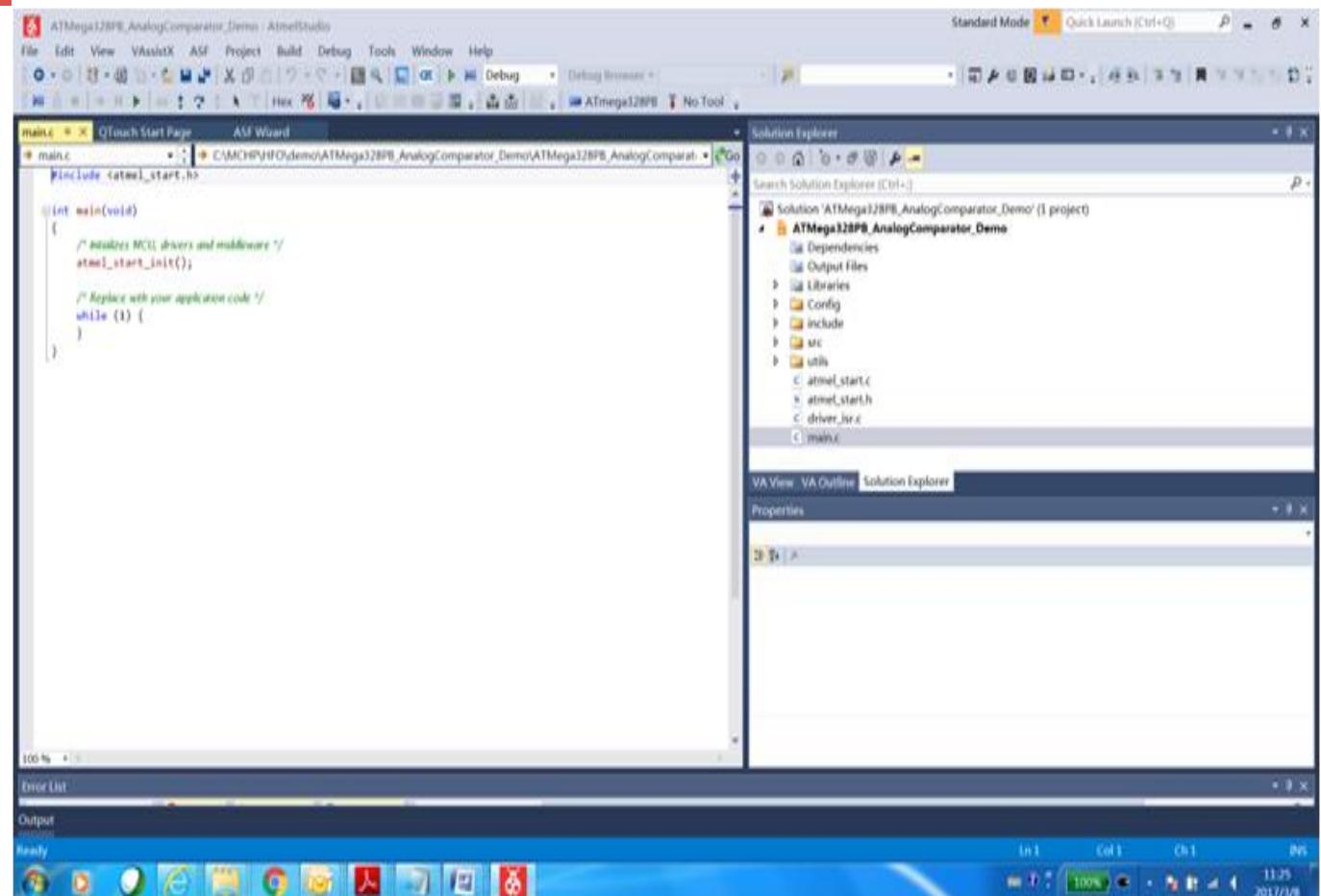
7 Configurador de reloj de configuración:



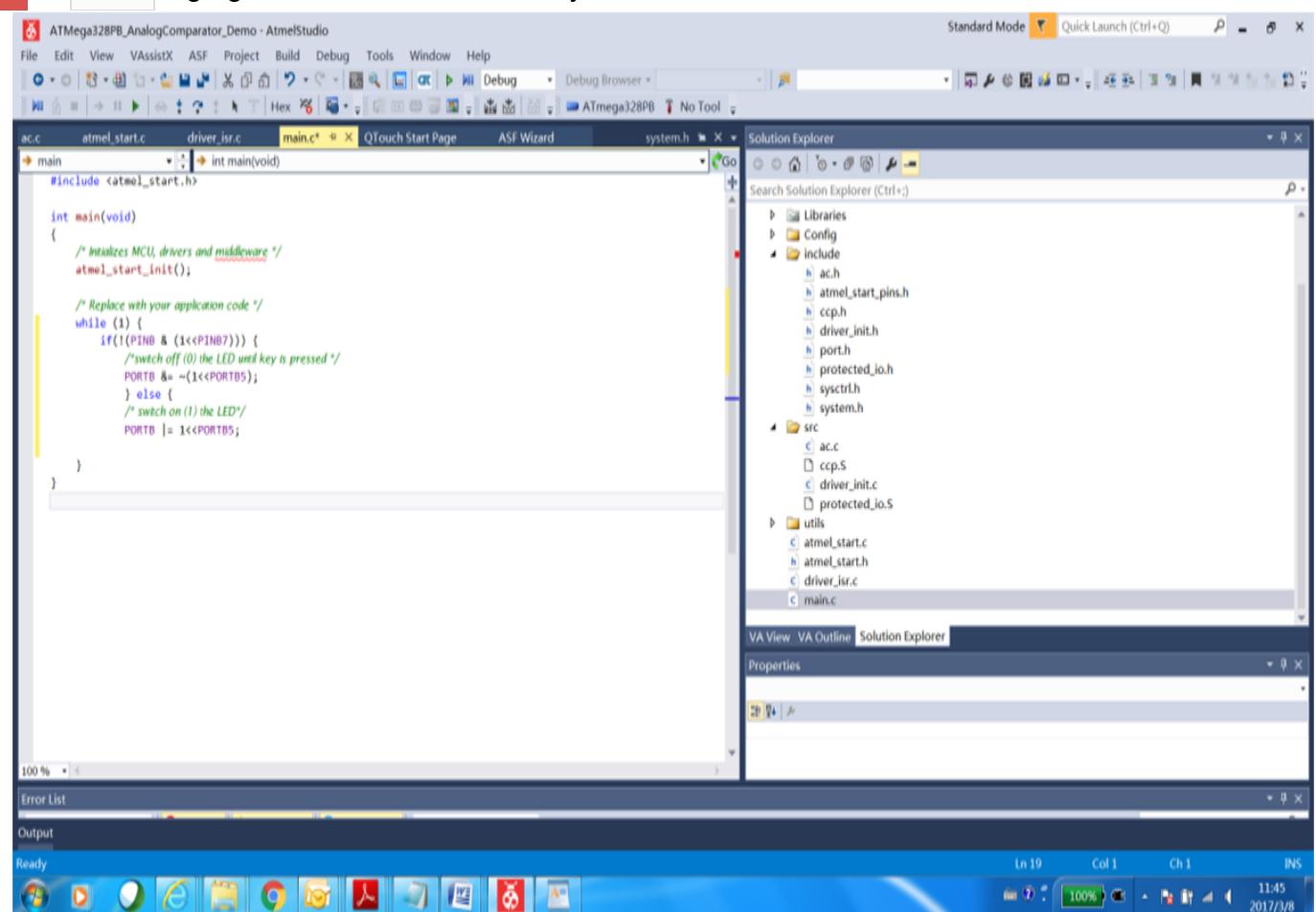
8 Generar proyecto:



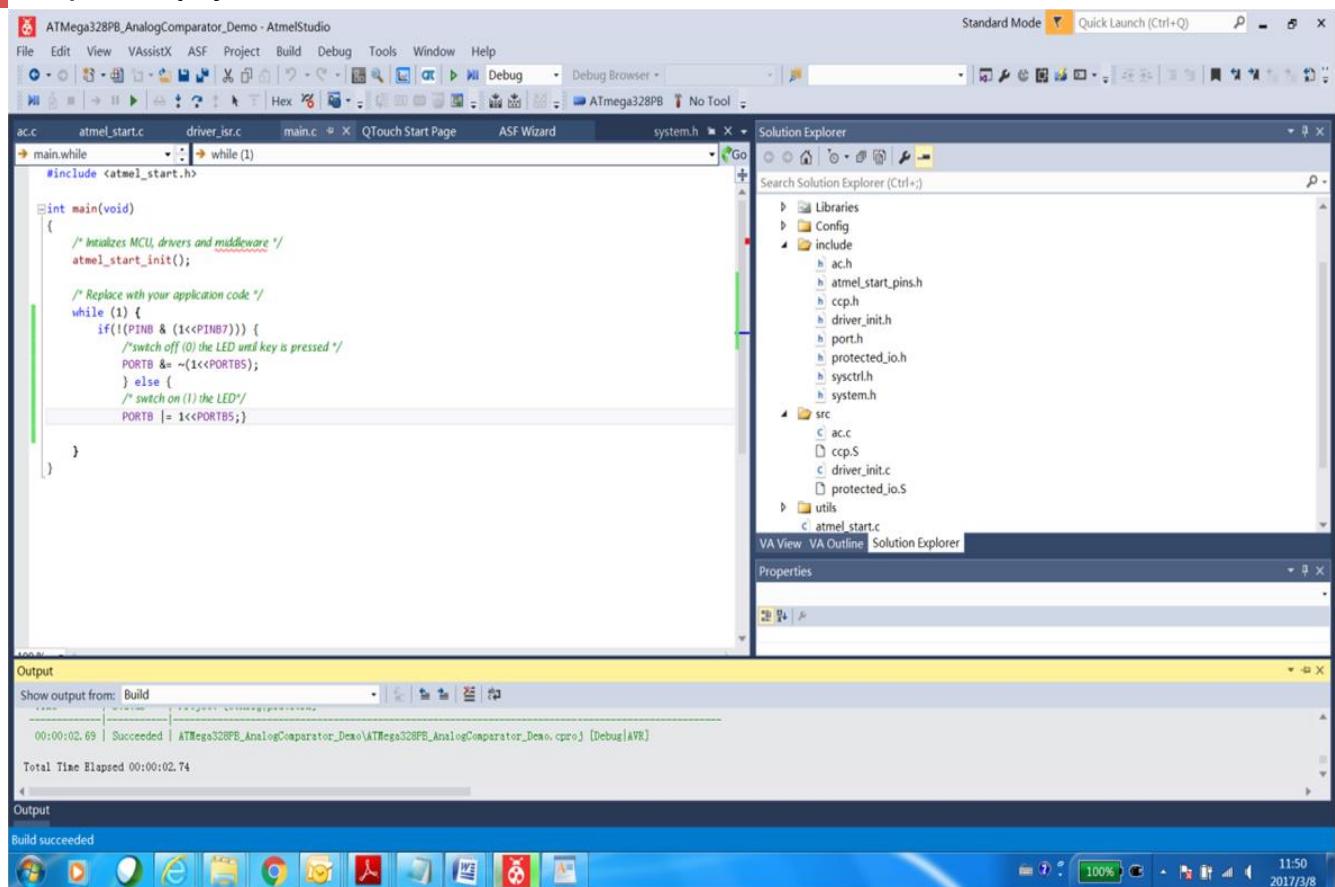
9 Ver proyecto:



10 En Main.c, agregue el escaneo del teclado y el controlador LED:



11 Compilación y ejecución:



Temporizador / Contadores

Descripción general del temporizador AVR® MCU

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

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 operación asincrónica

Temporizador 2 TC2 Temporizador/contador de 8 bits con PWM y operación asincrónica

Temporizador 3 TC3 Temporizador/contador de 16 bits con PWM y operación asincrónica

Temporizador 4 TC4 Temporizador/contador de 16 bits con PWM y operación asincrónica

Definiciones:

Nomenclatura de registros

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

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

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

TC0 - Temporizador/Contador de 8 bits con PWM

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

Registros TC0

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

Name: TCCR0B Offset: 0x45 Reset: 0x00 Property: When addressing as I/O Register: address offset is 0x25								
Bit	7	6	5	4	3	2	1	0
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Fuentes de temporizador/contador de reloj TC0

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

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

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

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

Unidad de contador T0

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

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

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

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

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

Modo	WGM2:0	Modo de	Arriba	Actualización	Bandera TOV
0	0 0 0	Normal	0xFF	Inmediato	Máximo
1	0 0 1	Fase PWM correcta	0xFF	Arriba	FONDO
2	0 1 0	Ctc	OCRA	Inmediato	Máximo
3	0 1 1	PWM rápido	0xFF	FONDO	Máximo
4	1 0 0	Reservado	~	~	~
5	1 0 1	Fase PWM correcta	OCRA	Arriba	FONDO
6	1 1 0	Reservado	~	~	~
7	1 1 1	PWM rápido	OCRA	FONDO	Máximo

Nota:

1.MAX = 0xFF

2. ABAJO = 0x00

Modos de funcionamiento para TC0

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

Los modos de operación disponibles para TC0 son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta

Borrar temporizador en el modo Comparar coincidencia

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

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

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

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

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

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

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

Registros (TC1, TC3, TC4)

- El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son registros de 16 bits.
- 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 (TIFR_n). Todas las interrupciones se enmascaran individualmente con el Registro de máscaras de interrupción del temporizador (TIMSK_n).

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

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

Name: TCCR1B, TCCR3B, TCCR4B
Offset: 0x81 + n*0x10 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
Access	ICNC	ICES		WGM3	WGM2		CS[2:0]	
Reset	0	0		0	0	0	0	0

Bits 2:0 – CS[2:0]: Clock Select [n = 0..2]

Los tres bits de Clock Select 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 preescalador)
0	1	1	clkio/64 (Del preescalador)
1	0	0	clkio/256 (Del preescalador)
1	0	1	clkio/1012 (Del preescalador)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en el borde descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en el borde 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 en dos ubicaciones de memoria de E/S de 8 bits: Counter High (TCNT_nH) que contiene los ocho bits superiores del contador y Counter Low (TCNT_nL) que contiene los ocho bits inferiores.

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

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

Modos de funcionamiento (TC1, TC3, TC4)

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

Los modos de operación disponibles son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta
- Modo PWM correcto de fase y frecuencia

Modo PWM rápido

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

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

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

$$f_{OCnxPWM} = \frac{f_{clk_I/0}}{N \cdot (1 + TOP)}$$

Nota:

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

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

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

Registros TC2

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

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

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

Fuentes de reloj TC2

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

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

Unidad de contador TC2

Dependiendo del modo de operación utilizado, el contador se borra, incrementa o disminuye en cada reloj temporizador (clkT2). clkT2 se puede generar a partir de una fuente de reloj externa o interna, seleccionada por los bits clock Select (CS2[2:0]).

La secuencia de conteo está determinada por la configuración de los bits WGM21 y WGM20 ubicados en el Registro de Control de Temporizador/Contador (TCCR2A) y el bit WGM22 ubicado en el Registro de Control de Temporizador/Contador B (TCCR2B).

Modos de funcionamiento de TC2

El modo de operación, es decir, el comportamiento del temporizador/contador y los pines de comparación de salida, se define mediante la combinación de los bits de modo de generación de forma de onda (WGM2[2:0]) y modo de comparación de salida (COM2x[1:0]).

Los modos de operación disponibles son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta

Modo normal

En el modo Normal (WGM22:0 = 0) la dirección de conteo siempre está arriba (incrementando) sin tener el contador despejado. El contador pasará a 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 a medida que el TCNT2 se convierta en cero. La bandera TOV2, en este caso, se comporta como un noveno bit, excepto que solo está configurada, no borrada. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el indicador TOV2, la resolución del temporizador se puede aumentar mediante software. No hay casos especiales a considerar en el modo Normal, se puede escribir un nuevo valor de contador en cualquier momento.

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

El temporizador debe iniciarse seleccionando la fuente del reloj.

Si se utilizan interrupciones, deben estar habilitadas.

Ejemplo de temporizador de MCU AVR® de 8 bits

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

Requisitos para ejecutar los ejemplos prácticos

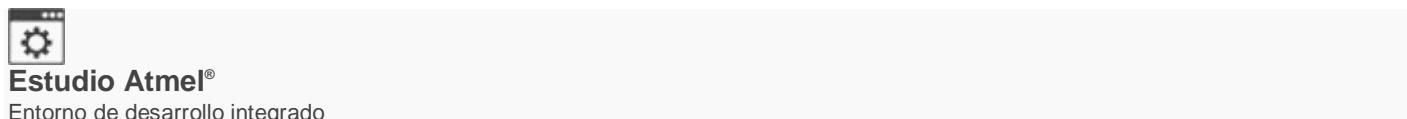
Requisitos de hardware

- ATmega328PB Xplained Mini placa
- Micro-USB Cable
- 
- **ATmega328PB Xplained Mini**
Kit de evaluación

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 integrado totalmente integrado a bordo.

Este ejercicio práctico demuestra cómo configurar los diferentes temporizadores.

Herramientas de software



Estudio Atmel®
Entorno de desarrollo integrado

Archivos adicionales

Guía del usuario de xplained Board.

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 operación asincrónica.
Temporizador 2	TC2	Temporizador/contador de 8 bits con PWM y operación asincrónica.
Temporizador 3	TC3	Temporizador/contador de 16 bits con PWM y operación asincrónica.
Temporizador 4	TC4	Temporizador/contador de 16 bits con PWM y operación asincrónica.

Definiciones:

Nomenclatura de registros

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

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

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

Temporizador 0 - Temporizador/Contador de 8 bits con PWM

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

Registros TC0

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

Name: TCCR0B
Offset: 0x45
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
Access	FOCOA	FOCOB			WGM02	CS02	CS01	CS00
Reset	0	0			0	0	0	0

Fuentes de temporizador/contador de reloj TC0

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

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

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

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

Unidad de contador TC0

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

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

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

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

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

Modo	WGM2:0	Modo de	Arriba	Actualización	Bandera TOV
------	--------	---------	--------	---------------	-------------

0	0 0 0	Normal	0xFF	Inmediato	Máximo
1	0 0 1	Fase PWM correcta	0xFF	Arriba	FONDO
2	0 1 0	Ctc	OCRA	Inmediato	Máximo
3	0 1 1	PWM rápido	0xFF	FONDO	Máximo
4	1 0 0	Reservado	~	~	~
5	1 0 1	Fase PWM correcta	OCRA	Arriba	FONDO
6	1 1 0	Reservado	~	~	~
7	1 1 1	PWM rápido	OCRA	FONDO	Máximo

Nota:

1. MAX = 0xFF

2. ABAJO = 0x00

Modos de funcionamiento para TC0

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

Los modos de operación disponibles para TC0 son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta

Borrar temporizador en el modo Comparar coincidencia

En el modo Borrar temporizador en comparación (WGM0[2:0]=0x2), el registro OCR0A se utiliza para manipular la resolución del contador: el contador se borra a CERO cuando el valor del contador (TCNT0) coincide con el OCR0A. El OCR0A define el valor superior para el contador, de ahí también su resolución.

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

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

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

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

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

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

Registros (TC1, TC3, TC4)

- El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son registros de 16 bits.
- 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 máscaras de interrupción del temporizador (TIMSKn).

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

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

Name: TCCR1B, TCCR3B, TCCR4B
Offset: 0x81 + n*0x10 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
Access	ICNC	ICES		WGM3	WGM2		CS[2:0]	
Reset	R/W	R/W		R/W	R/W	R/W	R/W	R/W

Bits 2:0 – CS[2:0]: Clock Select [n = 0..2]

Los tres bits de Clock Select 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 preescalador)
0	1	1	clkio/64 (Del preescalador)
1	0	0	clkio/256 (Del preescalador)
1	0	1	clkio/1012 (Del preescalador)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en el borde descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en el borde 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 en dos ubicaciones de memoria de E/S de 8 bits: Counter High (TCNTnH) que contiene los ocho bits superiores del contador y Counter Low (TCNTnL) que contiene los ocho bits inferiores.

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

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

Modos de funcionamiento (TC1, TC3, TC4)

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

Los modos de operación disponibles son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta
- Modo PWM correcto de fase y frecuencia

Modo PWM rápido

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

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

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

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

Nota:

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

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

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

Registros TC2

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

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

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

Fuentes de reloj TC2

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

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

Unidad de contador TC2

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

Modos de funcionamiento de TC2

El modo de operación, es decir, el comportamiento del temporizador/contador y los pines de comparación de salida, se define mediante la combinación de los bits de modo de generación de forma de onda (WGM2[2:0]) y modo de comparación de salida (COM2x[1:0]).

Los modos de operación disponibles son:

- Modo normal
- Borrar temporizador en el modo Comparar coincidencia (CTC)
- Modo PWM rápido
- Modo PWM de fase correcta

Modo normal

En el modo Normal (WGM22:0 = 0) la dirección de conteo siempre está arriba (incrementando), sin tener el contador despejado. El contador pasará a 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 a medida que el TCNT2 se convierta en cero. La bandera TOV2, en este caso, se comporta como un noveno bit, excepto que solo está configurada, no borrada. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el indicador TOV2, la resolución del temporizador se puede aumentar mediante software. No hay casos especiales a considerar en el modo Normal, se puede escribir un nuevo valor de contador en cualquier momento.

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

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

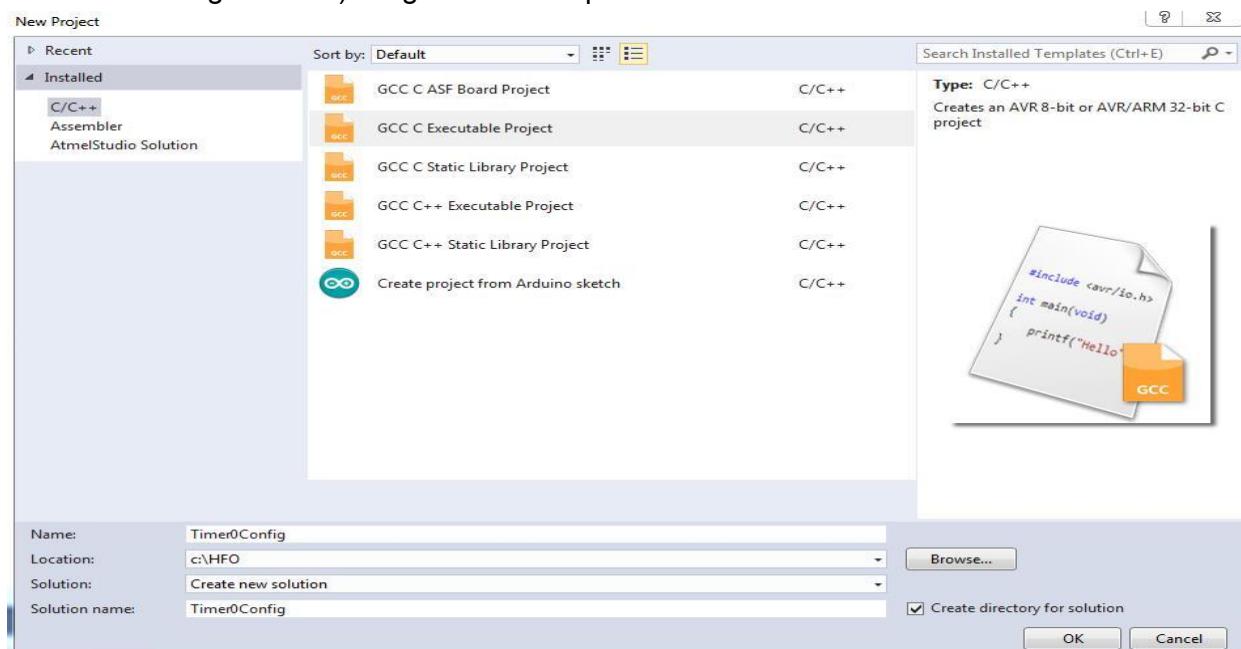
Paso a paso

Creación de proyectos

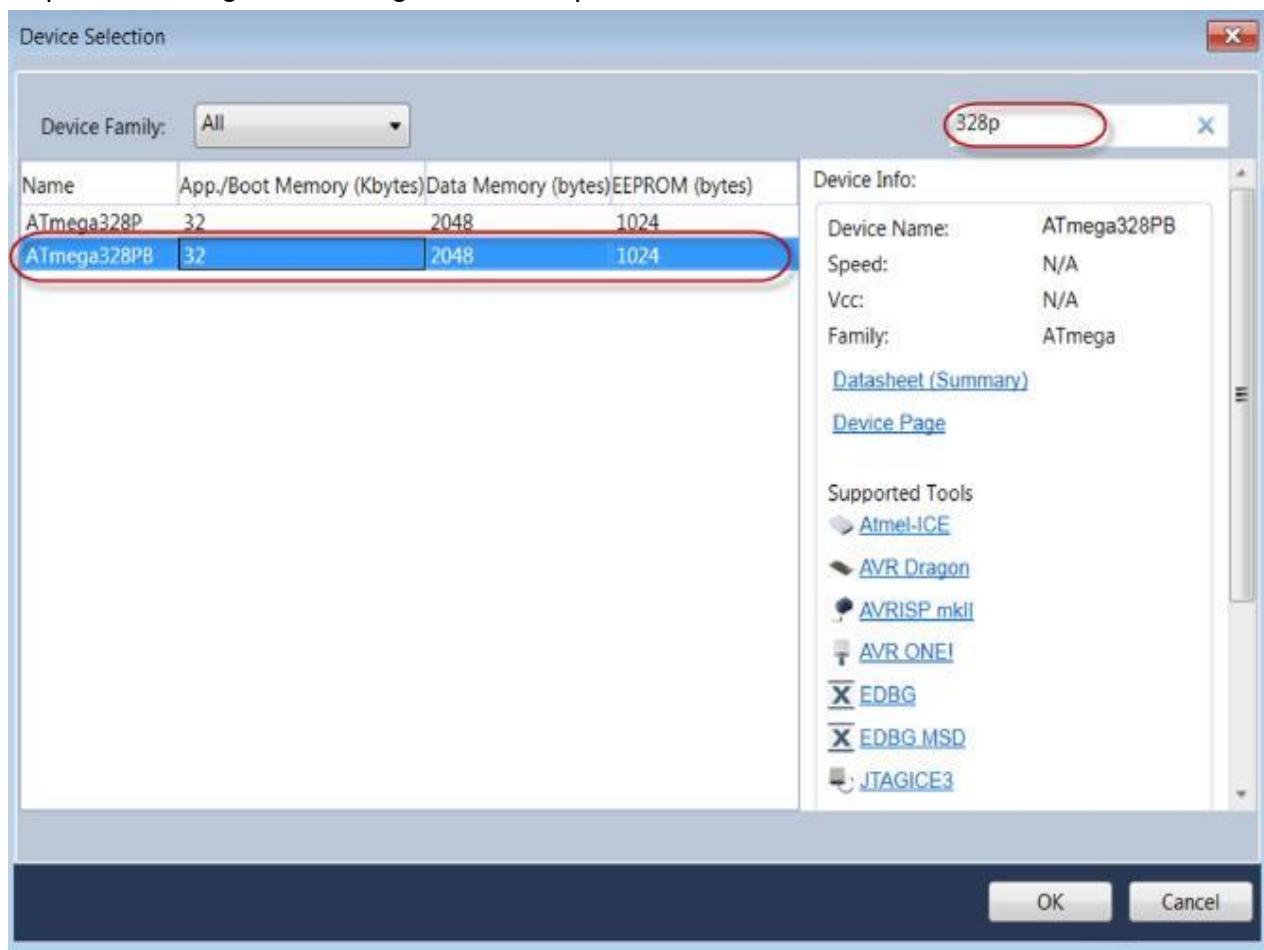
1 Abrir Atmel Studio 7

2 Seleccione archivo > nuevo proyecto de >

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



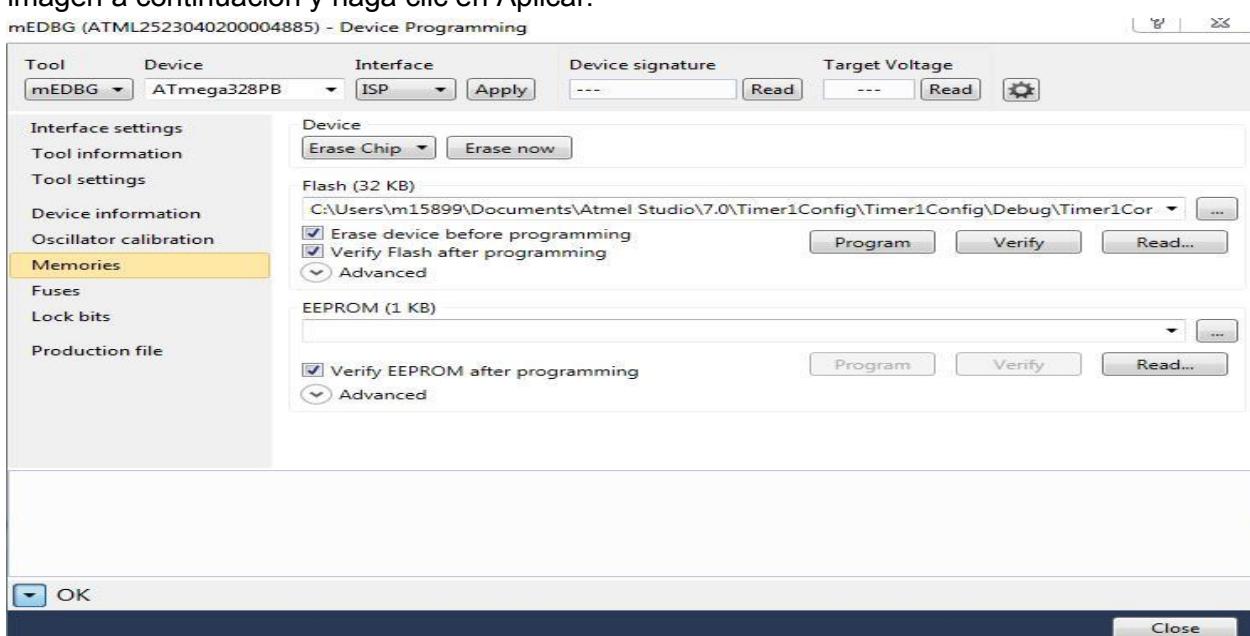
- 4 En la barra de búsqueda de la ventana Selección de dispositivos, escriba "328p" y luego seleccione el dispositivo Atmega328PB, haga clic en Aceptar.



Programación de la placa

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

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



7 Seleccione Memorias > programa Espere hasta que el botón "Verificando Flash... OK" para que aparezca.

Modificaciones del proyecto

avr/io.h y avr/interrupt.h deben ser "#included" 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 - Parpadear un LED usando TC0

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

A Modifique la función principal agregando el siguiente código:

```
1 int main(void)
2{
3    //set RB5 as output
4    DDRB |= 1 << DDBR5;
5
6    //call TMR0 initialization function
7    init_TC0();
8
9    //enable interrupt
10   sei();
11
12   while (1)
13   {
14       //main loop
15   }
16 }
```

B Cree la función init_TC0 () en main.c. Agregue el código siguiente:

```
1 void init_TC0(void)
2{
3
4    // Set the Timer Mode to CTC
5    TCCR0A |= (1 << WGM01);
6
7    // Set the value that you want to count to
8    OCR0A = 0xF9; //249
9
10   //Set the ISR COMPA vect
11   TIMSK0 |= (1 << OCIE0A);
12
13   // set prescaler to 1024 and start the timer
14   TCCR0B |= (1 << CS00) | (1 << CS02);
15 }
```

El código anterior hace lo siguiente:

- Configura el temporizador 0 en modo CTC estableciendo el bit WGM correspondiente en el registro TCCR0A;

- Establece el valor TOP en el registro OCR0A calculado utilizando la ecuación 1 para parpadear el LED a 30 kHz;
- Habilita la interrupción del temporizador;
- Establece el preescalador en 1024 configurando los bits CS00 y CS02 en el registro TCCR0B;

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

```
1 ISR (TIMER0_COMPA_vect)
2{
3     //event to be executed every 32ms*MyTimerConstant
4     counter++;
5     if (counter == MyTimerConstant) {
6         counter = 0;
7         PORTB ^= 1 << PORTB5; //toggle LED on PORTB5
8     }
9 }
```

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

9Modificación 2 - Uso de TC1 en modo PWM para atenuar el LED

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

AModifique `main()` agregando el siguiente código.

```
1 int main(void)
2
3 {
4     //set direction of pin PB1 set as output
5     DDRB |= 1 << DDB1;
6
7     init_TC1_pwm();
8
9     //enable global interrupts
10    sei();
11
12    while (1)
13    {
14        //main loop
15    }
16 }
```

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

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

```
1 void init_TC1_pwm(void)
2 {
3     //clear OCnA on compare match, BOTTOM (non-inverting mode)
4     TCCR1A = (1 << COM1A1);
5
6     //the counting sequence is determined by the setting of the waveform generation
7     mode bits in Timer
8     TCCR1A |= (1 << WGM10);
9     TCCR1B |= (1 << WGM12);
10
11    //256 prescaler clock select bits
12    TCCR1B |= (0 << CS10) | (1 << CS12);
13
14    //enable interrupts
15    TIMSK0 |= (1 << OCIE1A);
16}
```

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 Compare Output Mode, Fast PWM en el registro TCCR1A);
- Configura bits 1:0 – WGM[1:0]:Waveform Generation Mode 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 Parpadear un LED usando TC2

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

A Modifique la función principal agregando el siguiente código:

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

- Configura el bit de dirección del pin PB5, escribe el bit DDB5 en la lógica 1 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 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 bucle principal para ejecutar el evento LED parpadeante:

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

Configure el registro PORTB para que parpadee el LED

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

Sensor de temperatura interno

Sensor de temperatura interno AVR

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

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

Datos de medición de muestras

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

Calibración

Los resultados de las mediciones de temperatura tienen errores de compensación y ganancia. La referencia de temperatura interna se puede corregir para estos errores realizando mediciones de calibración a una o dos temperaturas conocidas y ajustando los valores de salida. Esto puede resultar en mediciones de temperaturas muy precisas, a veces tan precisas como ± 2 ° C. Se pueden encontrar más detalles en esta nota de aplicación.

Configuración del ADC

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

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

Una vez completada la conversión, el resultado se almacena en dos registros de datos ADC de 8 bits ADCH (8 bits más altos) y ADCL (8 bits inferiores). El resultado de 10 bits puede estar justificado a la izquierda o justificado a la derecha. Si el bit ADLAR se establece en un "1", entonces el resultado se deja

ajustado a los 10 bits superiores de los dos registros. Si se establece en "0", el resultado ocupa los 10 bits inferiores de los dos registros. De forma predeterminada, cada bit está borrado y la palabra está justificada correctamente.

Esta instrucción de código establecerá los bits como se describe.

$ADMUX = (1 \ll REFS1) | (1 \ll REFS0) | (0 \ll ADLR) | (1 \ll MUX3) | (0 \ll MUX2) | (0 \ll MUX1) | (0 \ll MUX0);$

Name: **ADMUX**

Offset: 0x7C

Reset: 0x00

Property: -

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

Bits 7:6 – REFSn: Reference Selection [n = 1:0]

These bits select the voltage reference for the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 29-3 ADC Voltage Reference Selection

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

Bits 3:0 – MUXn: Analog Channel Selection [n = 3:0]

The value of these bits selects which analog inputs are connected to the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in [ADCSRA](#) on page 323 is set).

Table 29-4 Input Channel Selection

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

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

Configuración del reloj ADC y el tiempo de conversión

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

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

Registro de control y estado de ADC A

Name: ADCSRA

Offset: 0x7A

Reset: 0x00

Property: -

Bit 7 – ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

Bit 6 – ADSC: ADC Start Conversion

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

Bit 5 – ADATE: ADC Auto Trigger Enable

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

Bit 4 – ADIF: ADC Interrupt Flag

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

Bit 3 – ADIE: ADC Interrupt Enable

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

Bits 2:0 – ADPSn: ADC Prescaler Select [n = 2:0]

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 29-5 Input Channel Selection

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

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

Iniciar una conversión

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

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

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

Objetivo

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

El sensor de temperatura en chip está acoplado a un solo canal ADC8 de extremo. Selección del canal ADC8 escribiendo `ADMUX`. `MUX[3:0]` a '1000' habilita el sensor de temperatura. La referencia de voltaje interna de 1.1 V también debe seleccionarse para la fuente de referencia de voltaje ADC en la medición del sensor de temperatura. Cuando el sensor de temperatura está habilitado, el convertidor ADC se puede utilizar en modo de conversión única para medir el voltaje sobre el sensor de temperatura. La sensibilidad de voltaje es de aproximadamente 1 mV / ° C, la precisión de la medición de temperatura es de ± 10 ° C.

Materiales

Herramientas de hardware (opcional)



ATmega328PB Xplained Mini

Kit de evaluación

Herramientas de software



Estudio Atmel®

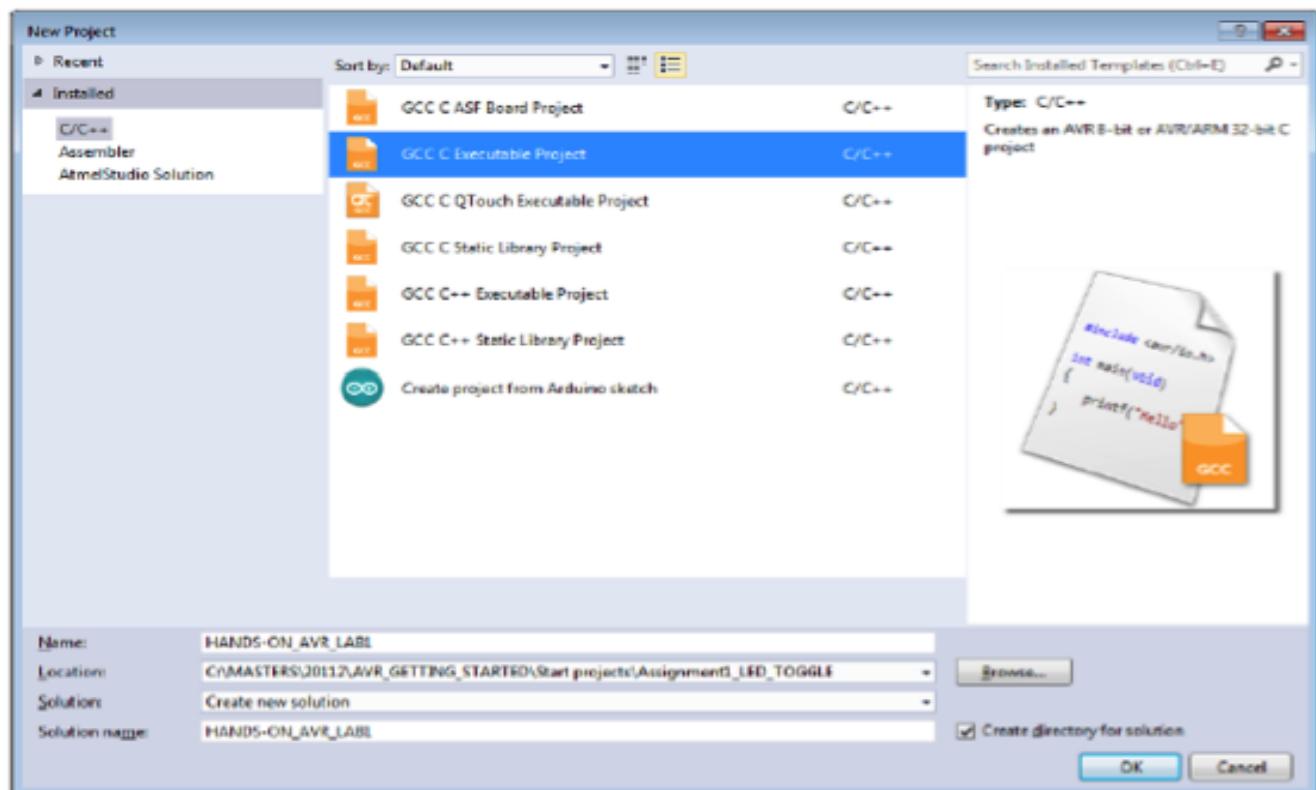
Entorno de desarrollo integrado

Archivo fuente `Main.c` <https://microchipdeveloper.com/install:example-and-exercise-files>

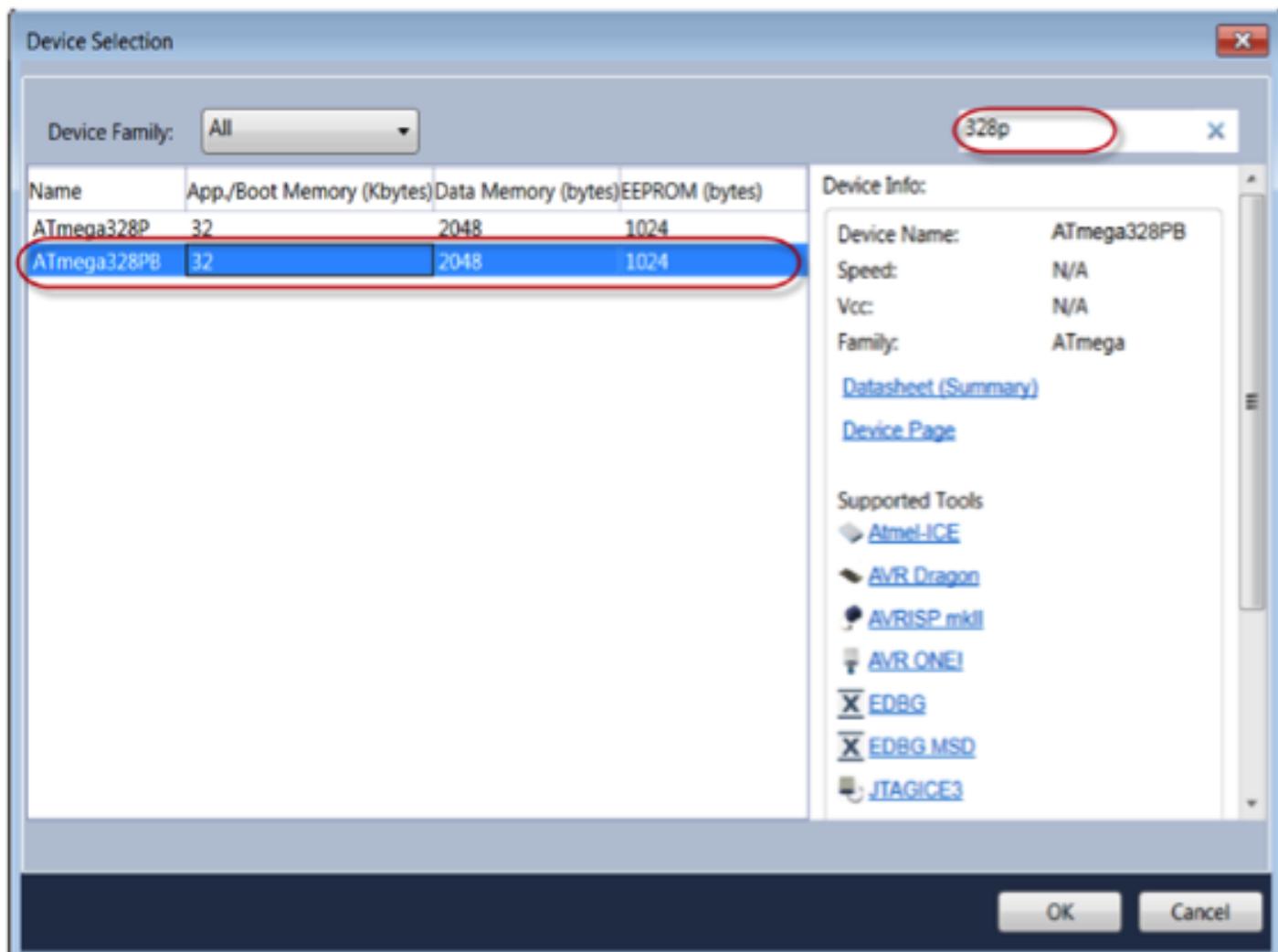
Procedimiento

1 Tarea 1 - Creación de proyectos

- Abrir Atmel Studio 7
- Seleccione archivo > nuevo proyecto de >
- Seleccione GCC C Executable Project y asigne el nombre Project1
- Elija una ubicación para guardar el proyecto en su computadora



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



2 Tarea 2 - Main.c

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

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

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

```
unsigned int Ctemp;
unsigned int Ftemp;

int main(void)
{
    /* Setup ADC to use int 1.1V reference
    and select temp sensor channel */
    ADMUX = (1<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (1<<MUX3) | (0<<MUX2) | (0<<MUX1) | (0<<MUX0);

    /* Set conversion time to
    112usec = [(1/(8Mhz / 64)) * (14 ADC clocks per conversion)]
    and enable the ADC*/
    ADCSRA = (1<<ADPS2) | (1<<ADPS1) | (1<<ADEN);

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

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

    /* Scan for changes on A/D input pin in an infinite loop */
}
```

```

while(1)

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

    ADCSRA |= (1<<ADSC);

    /* wait for conversion to complete */

    while ((ADCSRA & (1<<ADSC)) != 0)
    {

        /* Calculate the temperature in C */

        Ctemp = (ADC - 247)/1.22;

        Ftemp = (Ctemp * 1.8) + 32;

    }

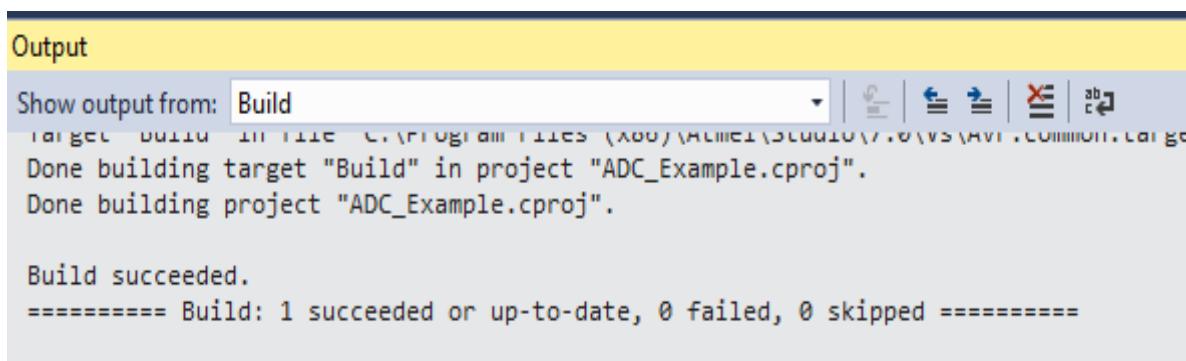
    return -1;
}

}

```

3 Tarea 3 - Generar proyecto

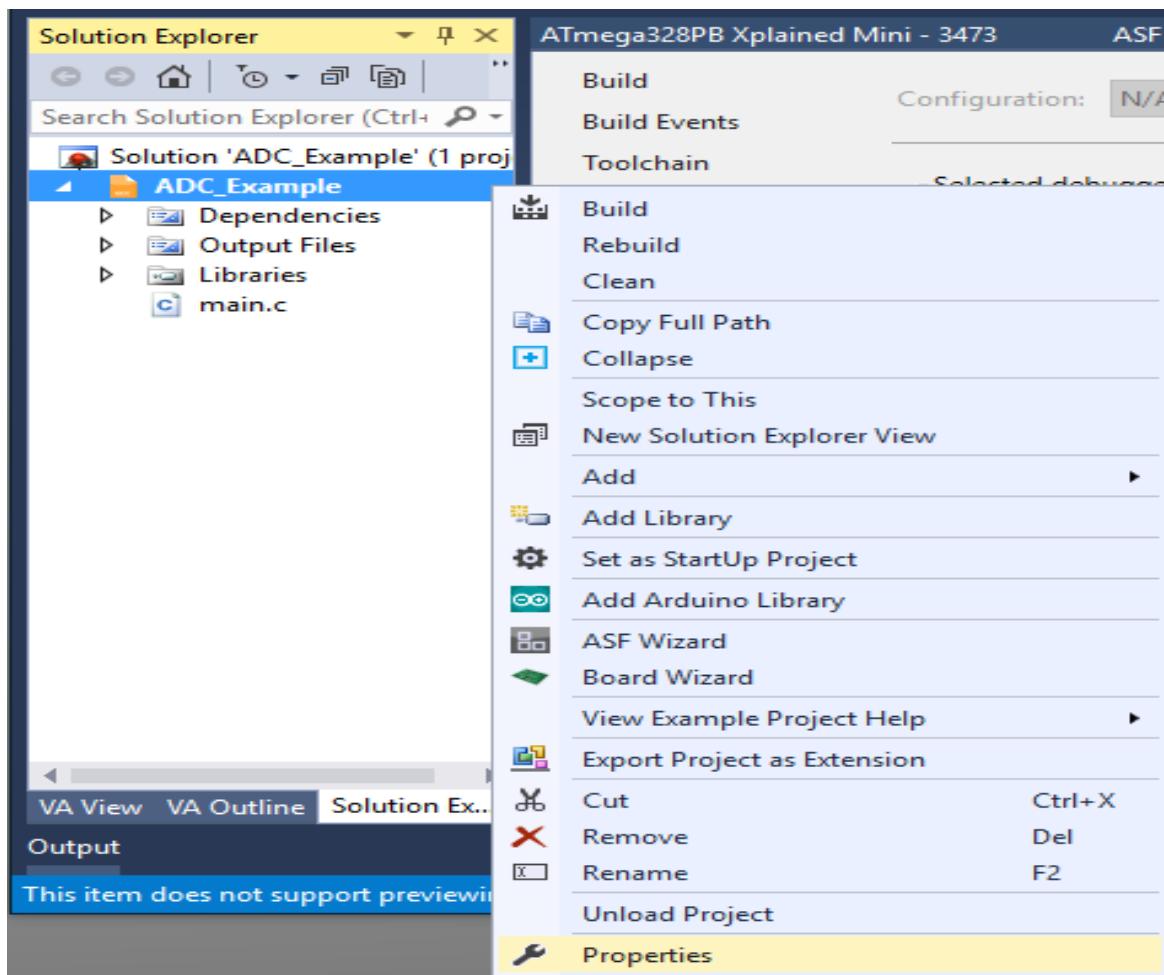
- Seleccione Generar > solución de compilación en el menú studio 7 para compilar el código. Verá un mensaje De compilación correcta en la ventana de salida. Si hay algún error, verifique main.c para ver si hay errores al ingresar el código del programa.



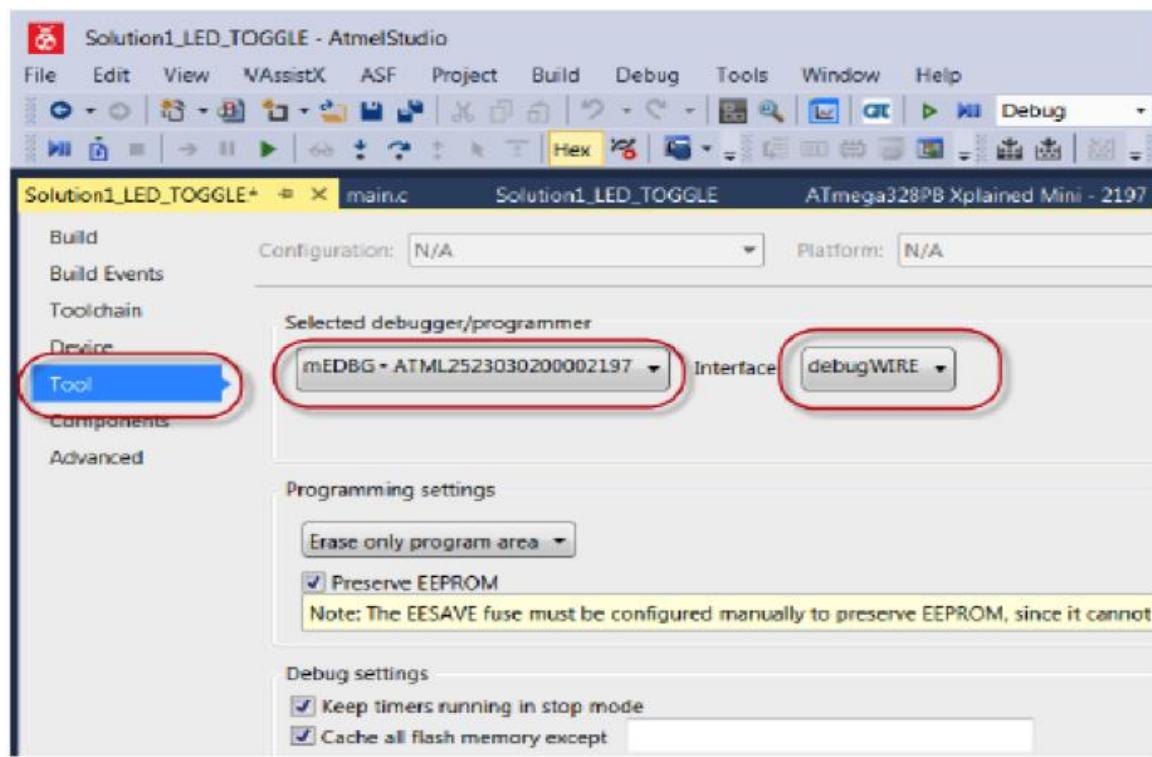
4 Tarea 4 - Programación de la placa Xplained

- Conecte la placa Xplained al puerto USB del ordenador mediante el cable incluido.

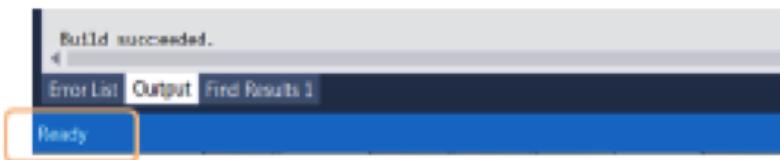
- En el área Explorador de soluciones, haga clic con el botón secundario en el nombre del proyecto y seleccione Propiedades.



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



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



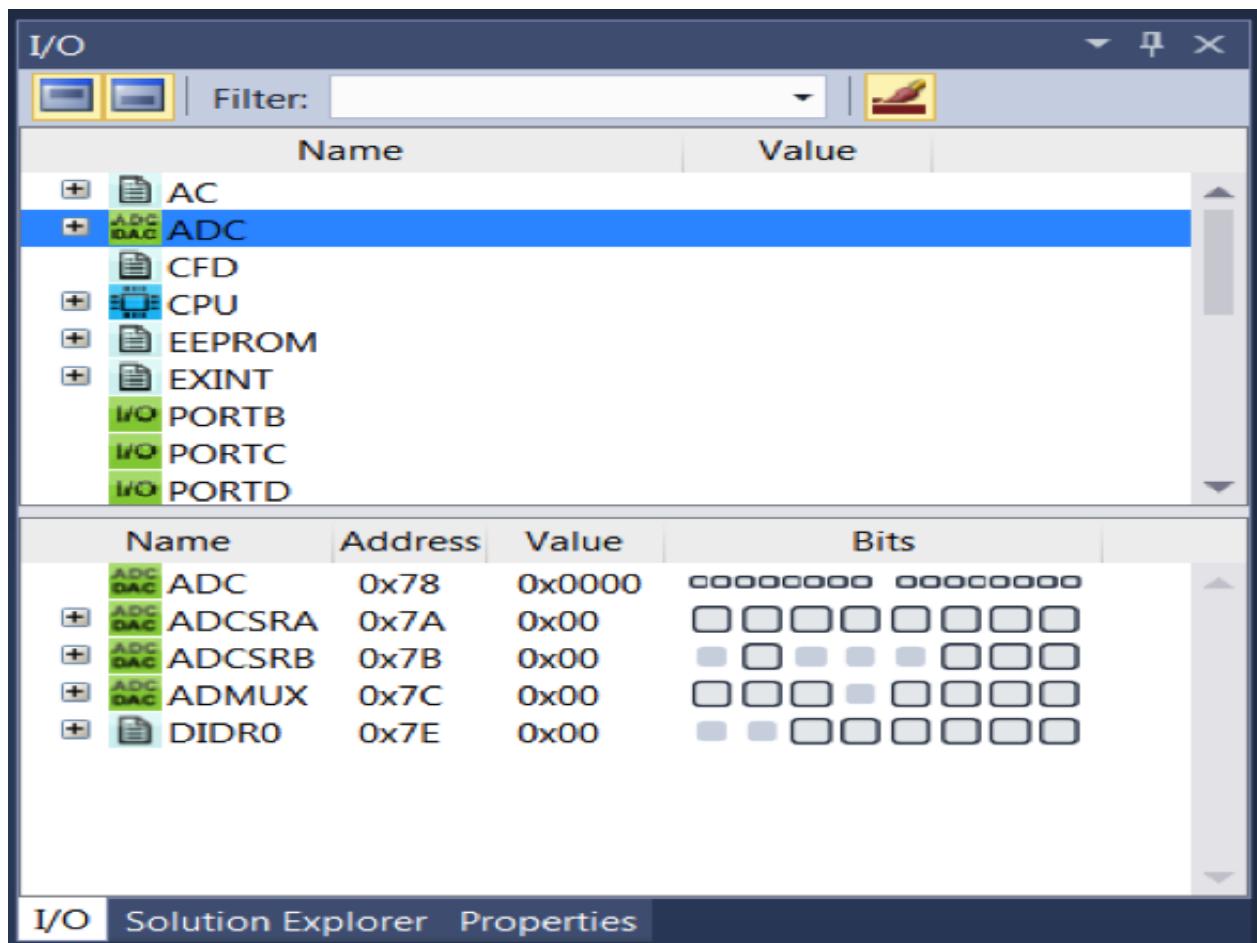
- Studio 7 mostrará un mensaje Listo cuando se complete la programación.

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

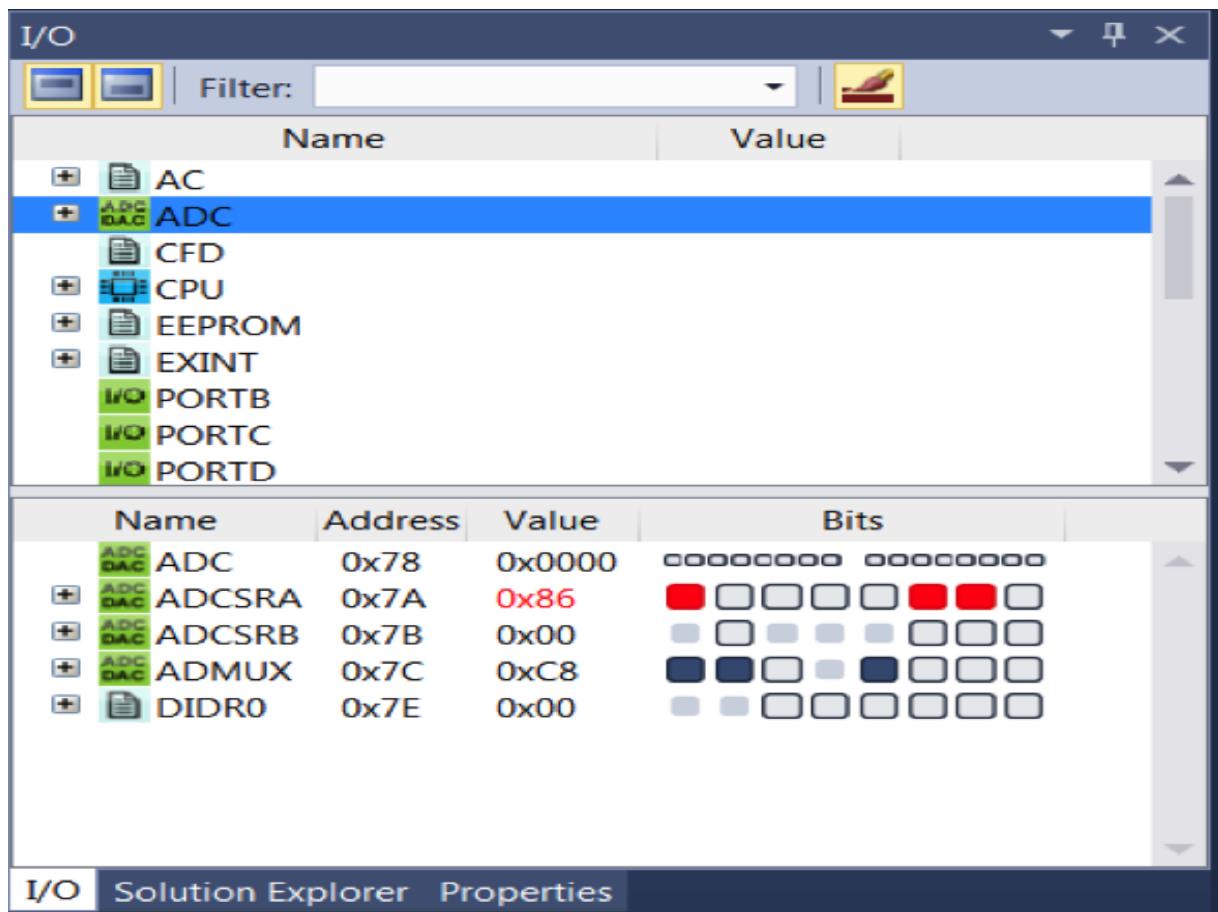
5 Tarea 5 - Depuración

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

- Seleccione Depurar > Iniciar depuración y Romper. El proyecto se construirá y el programa se cargará en la placa Xplained
- Se abrirá la ventana Vista de E/S que muestra los distintos periféricos
- Haga clic en la selección de ADC para abrir la vista de E/S para el ADC



- Seleccione Depurar > Paso a Paso (F10) a un solo paso a través del programa en la placa Xplained. Supervisar los registros de ADC en la vista de E/S mientras se realiza un solo paso



6 Tarea 6 - Punto de interrupción y salida

- Haga clic en el > 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 comandos en la instrucción ADCSRA dentro del bucle While

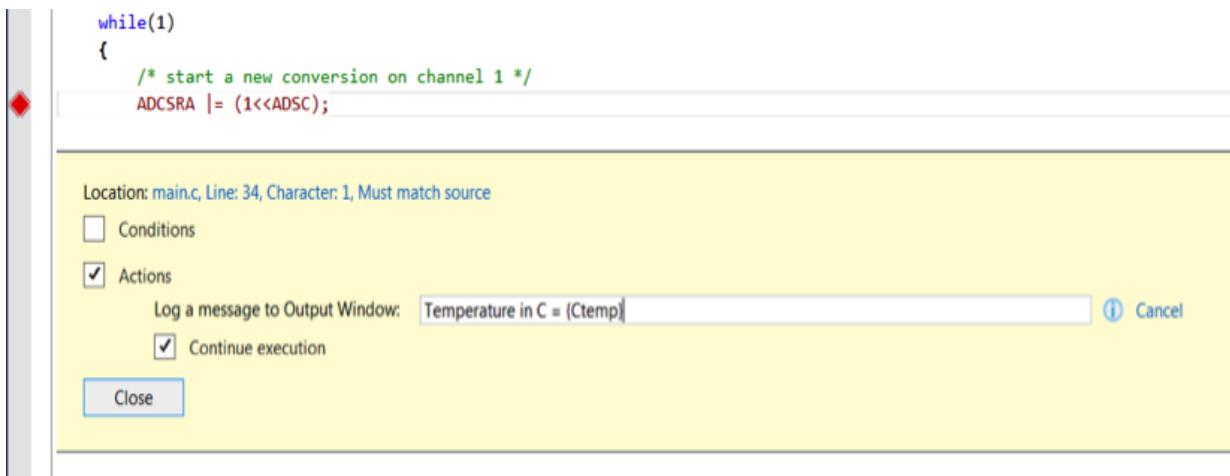
```

/* 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 1 */
    ADCSRA |= (1<<ADSC);
}

```

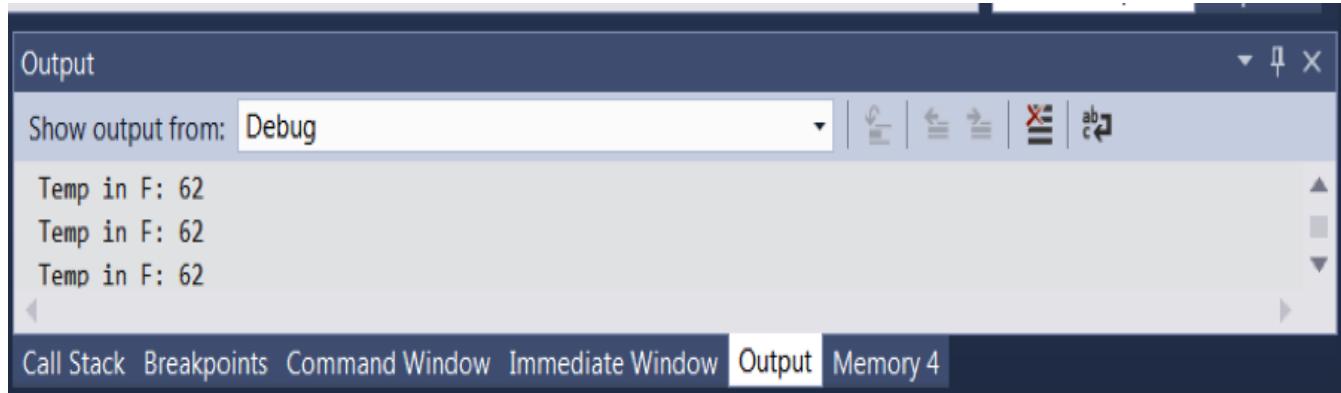
- Mueva el mouse sobre el círculo rojo del punto de interrupción, para poder ver la opción emergente de configuración (símbolo de engranaje) y haga clic en ella



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

Resultados

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



7 Tarea 7 - Deshabilitar debugWIRE y Cerrar

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

Análisis

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

Conclusiones

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

Operación de baja potencia

Descripción general del funcionamiento de baja potencia de AVR® MCU

Descripción general

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

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

El consumo de energía es proporcional al voltaje de funcionamiento, y para conservar la energía, debe considerar el uso de un voltaje del sistema lo más bajo posible. Además, el consumo también es directamente proporcional a la frecuencia del reloj, y si no se utilizan modos de suspensión, el dispositivo debe funcionar a la frecuencia más baja posible.

Consejos y trucos para reducir la potencia en un AVR®

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

Name: PRR0

Offset: 0x64

Reset: 0x00

Property: -

Bit 7 – 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.

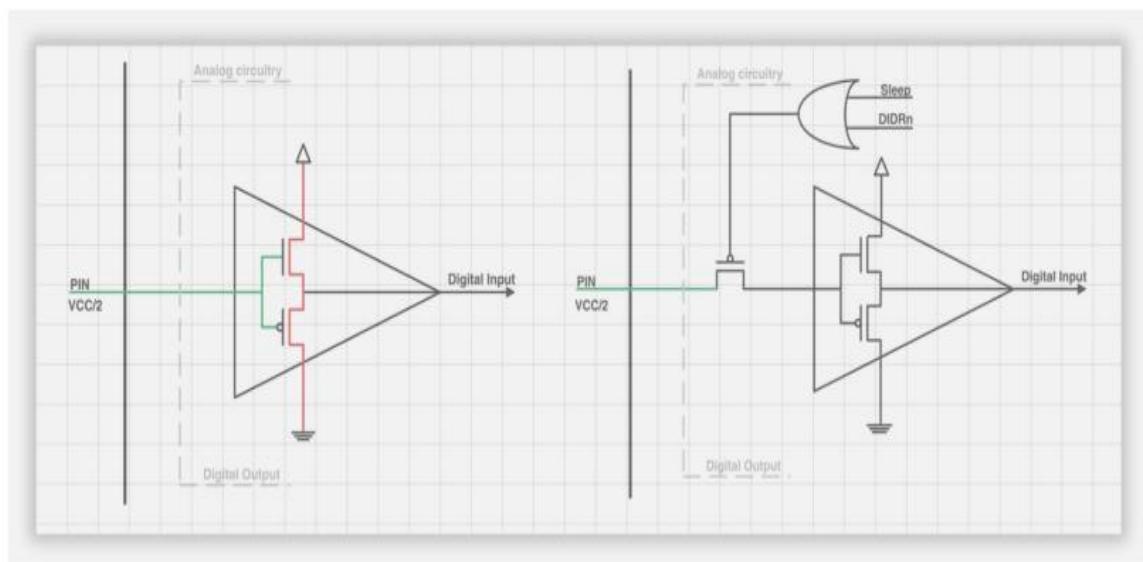
Bit 1 – PRUSART0: Power Reduction USART0

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.

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



Name: DIDR0

Offset: 0x7E

Reset: 0x00

Property: -

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

Name: DIDR1

Offset: 0x7F

Reset: 0x00

Property: -

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

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.

Proyecto de ejemplo de bajo consumo AVR

Objetivo

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

- Crear un proyecto y agregar código simple para parpadear un LED
- Ejecute el proyecto y mida el consumo de energía
- Realizar modificaciones en el código para reducir la potencia
- Ejecute el proyecto modificado y observe un consumo de energía reducido

Materiales

Requisitos de software



Estudio Atmel®

Entorno de desarrollo integrado

Requisitos de hardware

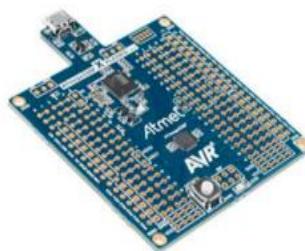
- ATmega328PB Xplained Mini placa
- Kit de depurador de energía
- Dos cables micro USB
- Tres alambres de hembra a macho, y un alambre de macho a macho



ATmega328PB Xplained Mini
Kit de evaluación

ATmega328PB Xplained Baord

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



Kit de depurador de energía

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



El depurador de potencia tiene dos canales de detección de corriente independientes para medir y optimizar el consumo de energía de un diseño.

El canal 'A' es el canal recomendado para medir con precisión las corrientes bajas. El canal 'B' es el canal recomendado para medir corrientes más altas con menor resolución.

Configuración de hardware

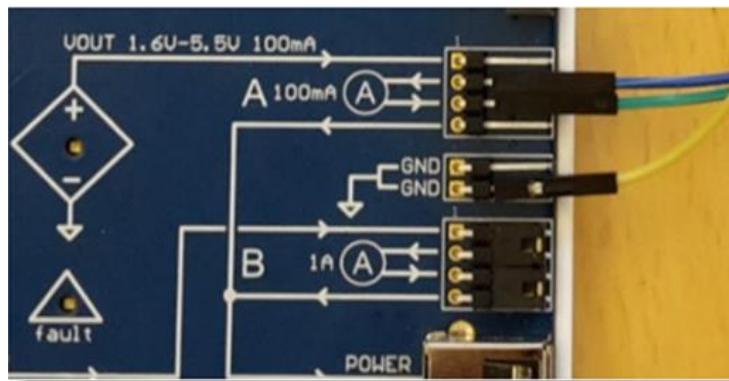
Conexión del depurador de energía a la placa Xplained ATmega328PB.

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

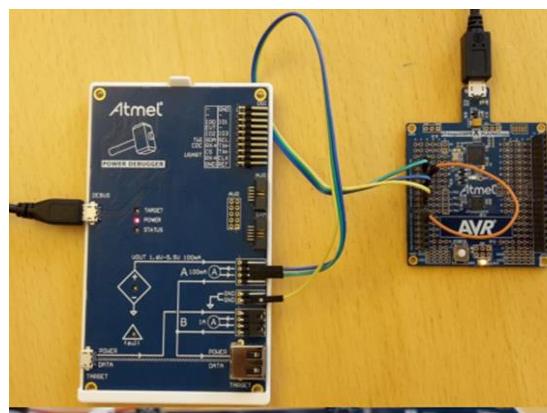
Tabla 1-1 Power Debugger y ATmega328PB Xplained Mini conexión.

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

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



Configuración de hardware

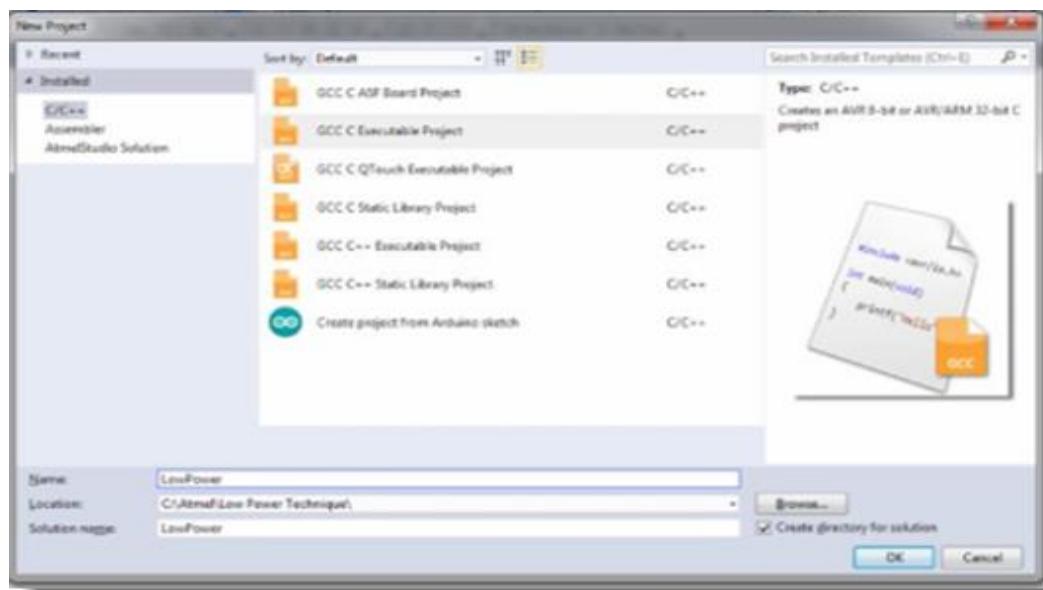


Medición de la corriente en modo activo

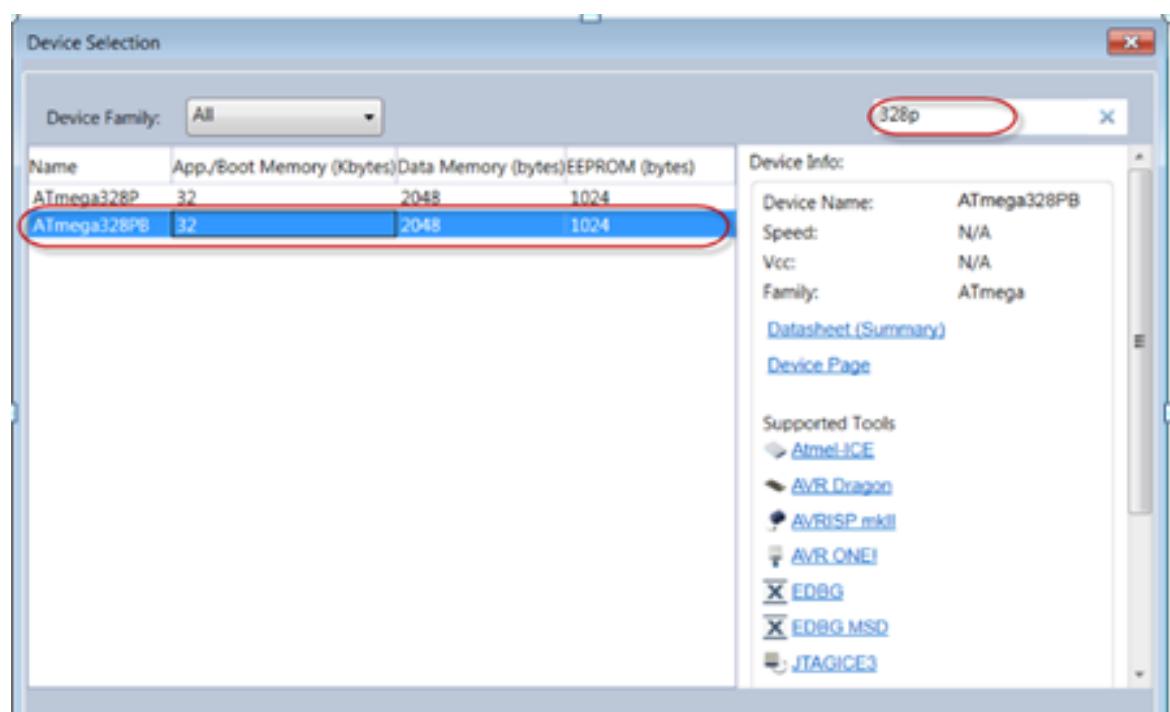
Procedimiento

A Creación de proyectos

- Abrir Atmel Studio 7
- Seleccione archivo > nuevo proyecto de >
- Seleccione GCC C Executable Project y asignele el nombre LowPower.
- Elija una ubicación para guardar el proyecto en el equipo.



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



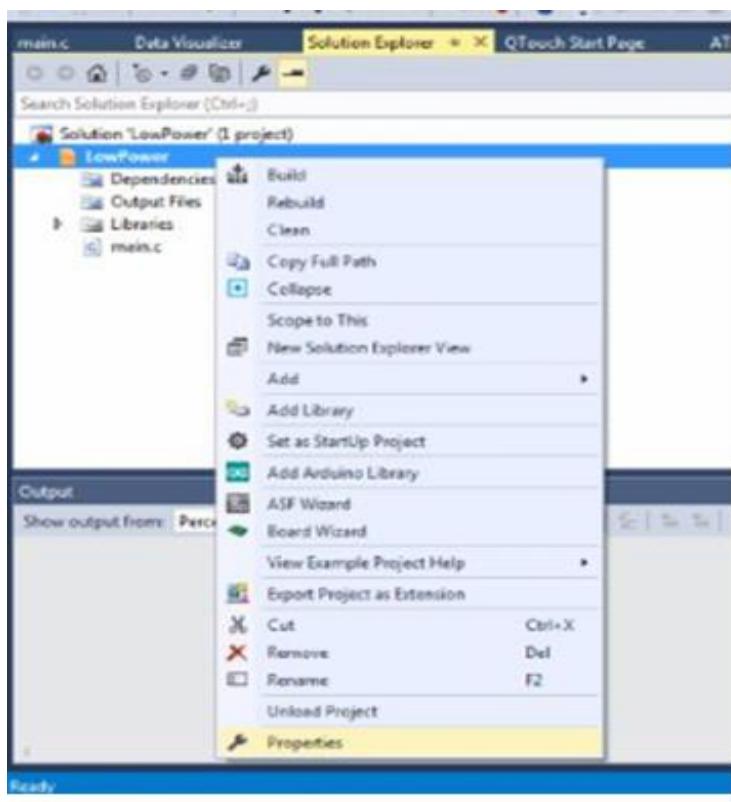
- Se creará un proyecto que contenga un bucle while(1) vacío en main().

```

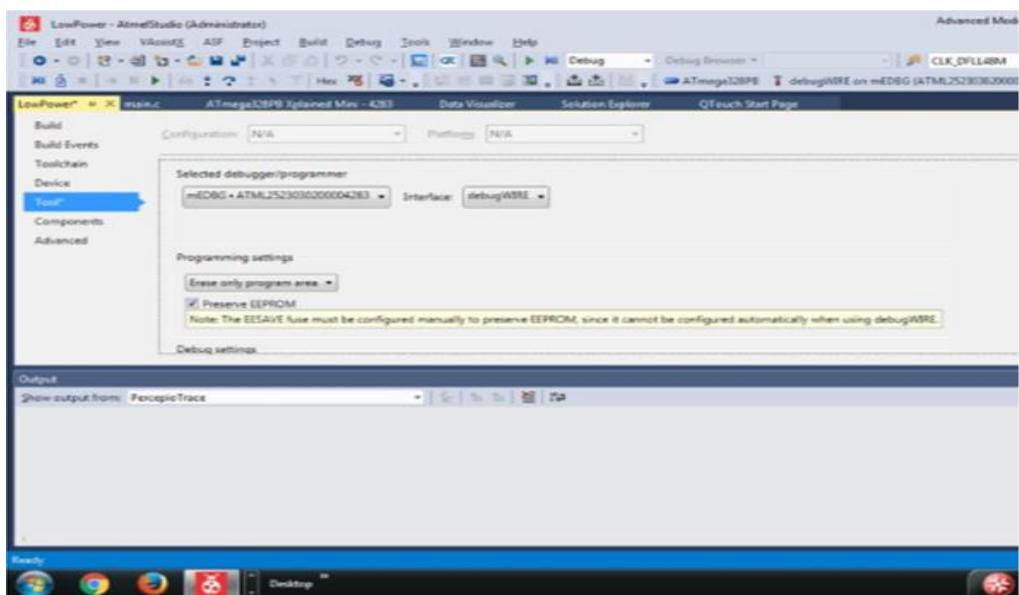
1 #include <avr/io.h>
2
3 int main(void)
4 {
5     /* Replace with your application code */
6     while (1)
7     {
8     }
9 }
```

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

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



En Herramienta, seleccione mEDBG y depurarWIRE



B Agregar código al proyecto

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

```
TMR0_init();  
  
DDRB |= 1<<DDRB5; // Direction of pin PB5 set as Output
```

Agregar la rutina de inicialización Timer 0

```
*****  
***** TMR 0 Initialization  
*****  
void TMR0_init( void ){  
  
    // enable timer overflow interrupt for both Timer0 and Timer1  
    TIMSK0=(1<<TOIE0)  
    // set timer0 counter initial value to 0  
    TCNT0=0x00;  
    // start timer0 with /1024 prescaler  
    TCCR0B = (1<<CS02) | (1<<CS00);  
  
    // enable interrupts  
    sei();  
}
```

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;  
        // Toogle LED  
        PORTB=PORTB ^ 0x20;  
    }  
}
```

El programa completo es el siguiente,

```
#include <avr/io.h>  
#include "avr/interrupt.h"  
  
    uint8_t      count;  
/* Timer 0 Interrupt */  
ISR(TIMER0_OVF_vect){  
  
    count++;  
    if(count==20){  
        count=0;  
        // Toogle LED  
        PORTB=PORTB ^ 0x20;  
    }  
}  
  
int main(void)  
{  
    TMR0_init();  
  
    DDRB |= 1<<DDRB5; // Direction of pin PB5 set as Output  
    /* Replace with your application code */  
    while (1)
```

```

}

/*****
    TMR 0 Initialization
 ****/
void TMRO_init( void ) {

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

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

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

    // enable interrupts
    sei();

}

```

C Comprobar que el proyecto se ejecuta

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 haya finalizado seleccionando Depurar > Deshabilitar debugWire y, a continuación, Cerrar.

D Medir el consumo de energía en modo activo

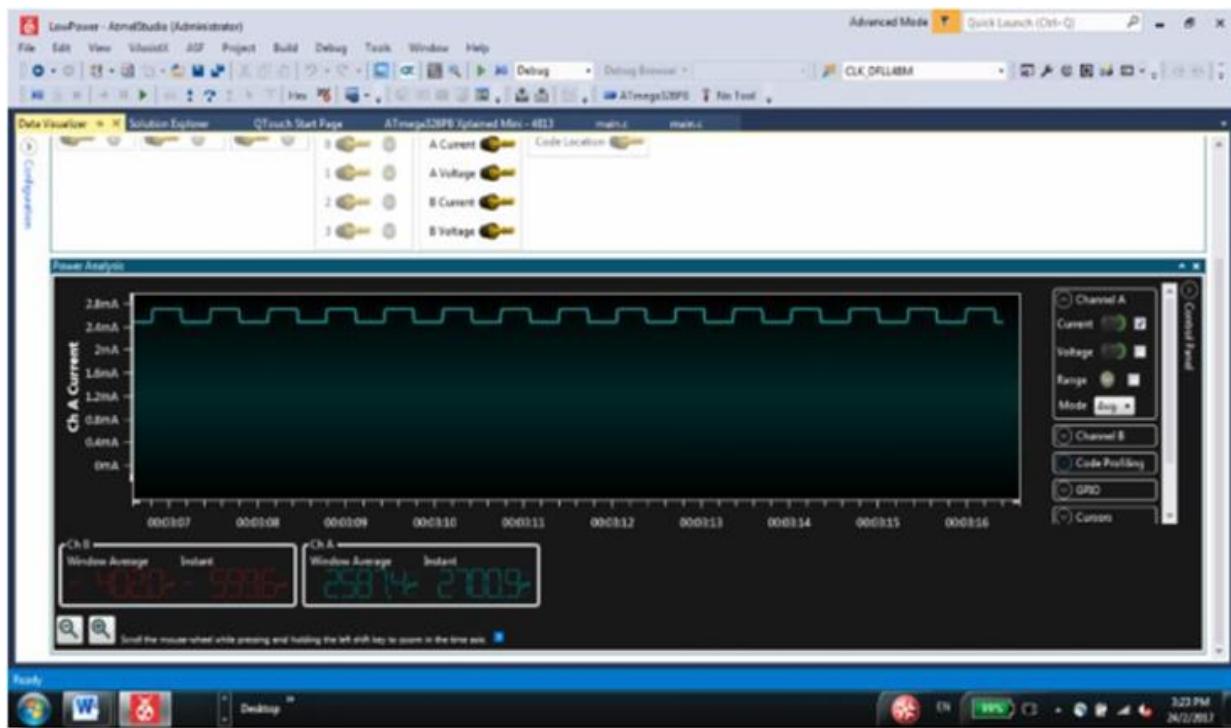
- En Atmel Studio 7, abra el menú Herramientas > Data Visualizer

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

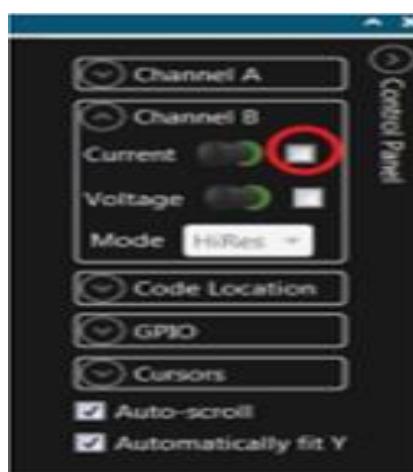


- Seleccione Encendido y arranque





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



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

Reducción del consumo de energía

Hay varios métodos que puede utilizar para reducir el consumo de energía de un microcontrolador AVR.

En este ejemplo se reduce la potencia en:[®]

desactivar periféricos no utilizados

Detener la corriente de fuga en los pines de E/S digitales

- Uso del modo de suspensión. Las técnicas de optimización del consumo de energía se implementan en la función `Optimize_power_consumption()` que se llama desde `main()`.

Deshabilitar los búferes de entrada digital y el comparador analógico

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

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

Desactivar periféricos no utilizados

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

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

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

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

```
/* Power shutdown to unused peripherals*/
```

```
PPR0 = 0xDF;
```

```
PPR1 = 0x3F;
```

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

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

4. Agregue el siguiente código en `optimize_power_consumption()` para desactivar el temporizador del perro guardián.

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

Aplicar resistencias pull-up

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

```
/* Unused pins set as input pull up*/  
DDRB &= 0xE0;  
DDRC &= 0xC9;  
DDRD &= 0x03;  
DDRE &= 0xF3;  
  
PORTB |= ~ (0xE0);  
PORTC |= ~ (0xC9);  
PORTD |= ~ (0x03);
```

```
PORTE |= ~(0xF3);
```

Usar la función de suspensión

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

1. Incluya `<avr/sleep.h>` archivo de encabezado de la biblioteca AVR en la parte superior del archivo.
2. Configure el modo de suspensión deseado en la función `optimize_power_consumption()` configurando el microcontrolador para que utilice el modo de suspensión `SLEEP_MODE_PWR_DOWN` para un consumo de energía mínimo.

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

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

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

Uso de la interrupción de cambio de pin

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

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

Configuración del bit PCINT7 y el registro PCICR:

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

Agregue la rutina de servicio de interrupciones:

Para habilitar las rutinas ISR, el nombre vectorial para el PCINT7 es ISR (PCINT0_vect), definido en el archivo de encabezado del dispositivo.

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

```
}
```

Código completado

```
/*
 * 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(void) {

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

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

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

    PORTB |=~(0xE0);
    PORTC |=~(0xC9);
    PORTD |=~(0x03);
    PORTE |=~(0xf3);

    /*Watchdog Timer OFF*/

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

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

}

***** TMR 0 Initialization *****
***** void TMR0_init( void ){
```

```

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

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

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

// enable interrupts
sei();

}

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

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

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

int main(void)
{
    TMR0_init();

    DDRB |= 1<<DDRB5;    // Direction of pin PB5 set as Output
    DDRB &= ~ (1<<DDB7); //Set PORTB7 as input

    optimize_power_consumption();

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

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

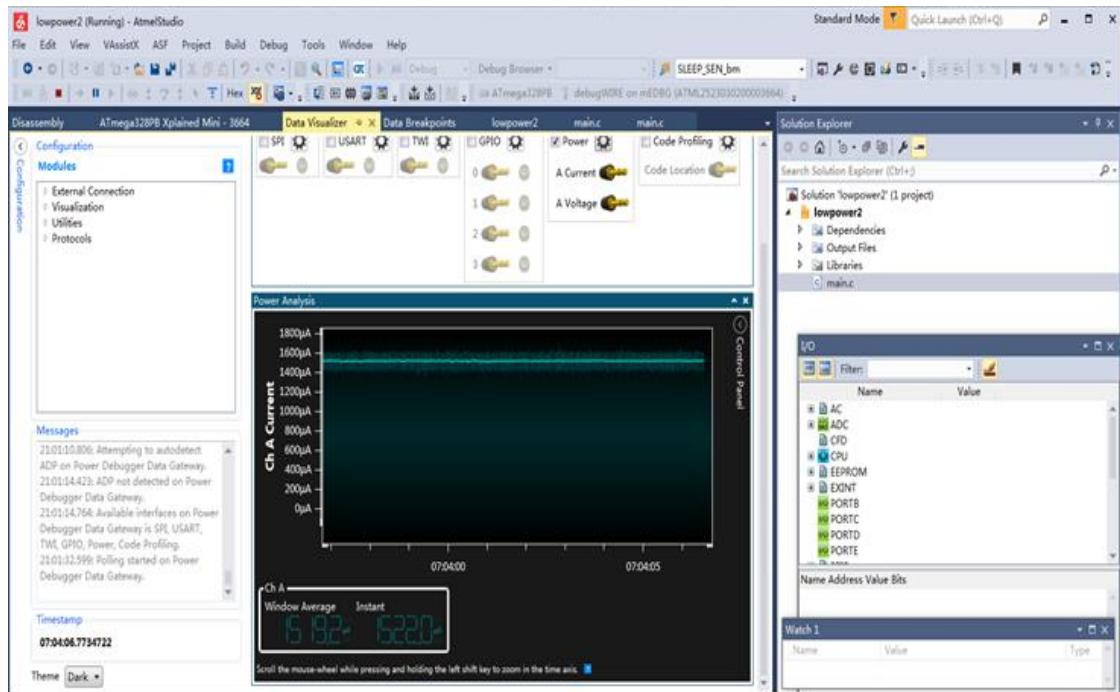
    /* Replace with your application code */
    while (1)
    {
        sleep_mode();
    }
}

```

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

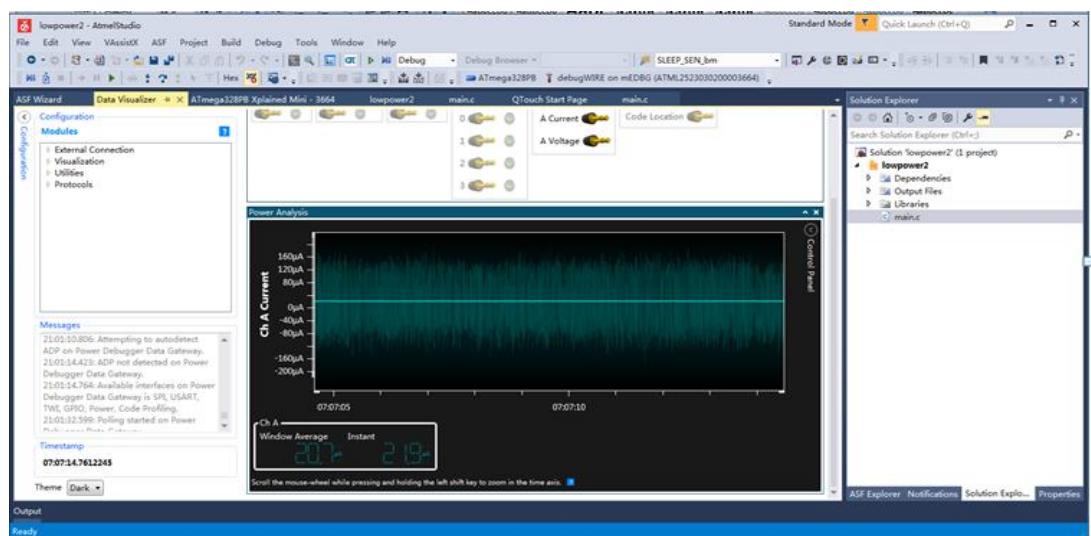
Programe la aplicación y mida el consumo de energía.

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 de DataVisualizer y verifique el consumo de energía como se muestra a continuación.



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

- Apagar el interruptor de alimentación de destino: Herramientas > configuración de Programación de dispositivos > Herramientas



El consumo actual debería haberse reducido a aproximadamente 20,7 μ A.

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

Conclusiones

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

- Diseño inteligente
- Uso de modos de suspensión
- Apagar periféricos no utilizados

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

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

Introducción:

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

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

Los proyectos de solución para cada una de las asignaciones están disponibles en el explorador de ejemplo START de Atmel.

En la categoría "Introducción", busque ADC y Power Optimization Solution (1-5). Los enlaces directos a los proyectos de ejemplo relevantes se proporcionan en las descripciones de las asignaciones a continuación.

- ADC y optimización de energía: manual tutorial práctico

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 de software

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

Tiempo estimado de finalización: 120 minutos

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

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

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

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

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

En esta asignación, el búfer de entrada digital en los pines de E/S se desactiva para reducir el consumo de corriente. El consumo de corriente se reduce aún más cuando el pin USART TX se configura como un pin de alta impedancia durante ningún período de transmisión de datos. Aquí se utilizan los mismos controladores y configuraciones de la asignación anterior. Atmel Studio se utiliza para desarrollar aún más el código.

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

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

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

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

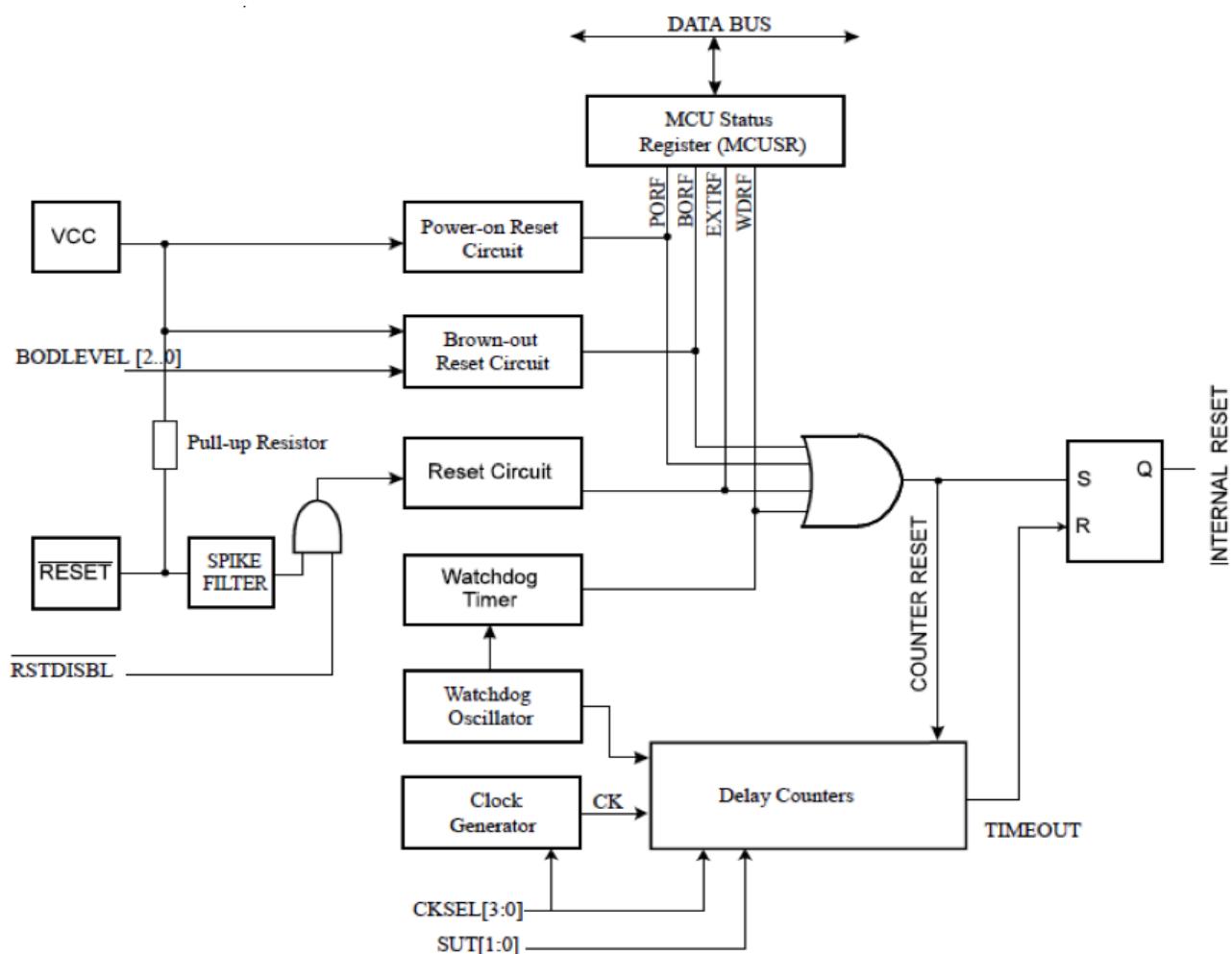
Resumen

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

Fuentes de restablecimiento de AVR

El dispositivo AVR tiene cuatro fuentes de restablecimiento:

- **Restablecimiento de encendido:** el microcontrolador (MCU) se restablece cuando el voltaje de alimentación es inferior al umbral de restablecimiento de encendido (VPOT).
- **Restablecimiento externo:** el MCU se restablece cuando hay un nivel bajo en el pin RESET durante más tiempo que la longitud mínima del pulso.
- **Restablecimiento del sistema Watchdog:** el MCU se restablece cuando expira el período del temporizador Watchdog y el modo Watchdog System Reset está habilitado.
- **Restablecimiento de apagado:** el MCU se restablece cuando la tensión de alimentación V_{CC} es menor que el Brown-out Reset.



Registro de estado de MCU (MCUSR)

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

Name: MCUSR

Offset: 0x54

Reset: 0x00

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

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

Bit 3 – WDRF: Watchdog System Reset Flag

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

Bit 2 – BORF: Brown-out Reset Flag

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

Bit 1 – EXTRF: External Reset Flag

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

Bit 0 – PORF: Power-on Reset Flag

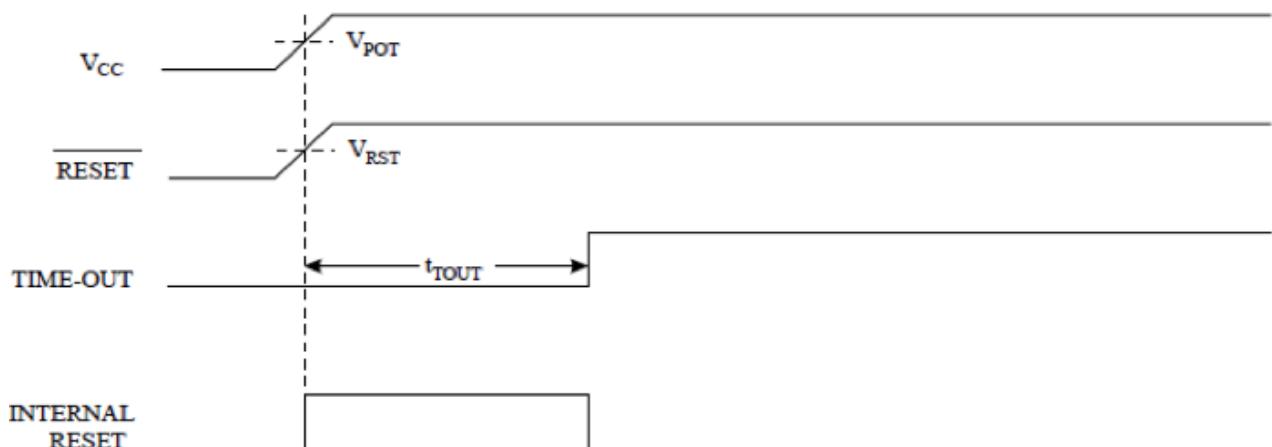
This bit is set if a Power-on Reset occurs. The bit is reset only by writing a '0' to it.

Restablecimiento de encendido (POR)

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

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

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

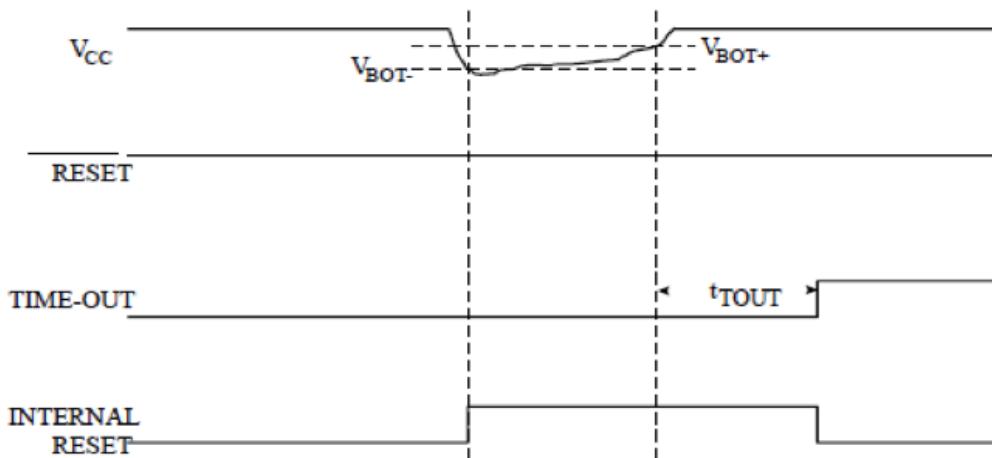


Detección de apagado (BOD) y restablecimiento de apagado (BOR)

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

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

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



Temporizador watchdog (WDT)

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

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

Detección de fallos de reloj (CFD)

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

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

Proyecto de ejemplo de restablecimiento de orígenes de AVR®

Objetivo

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

Para obtener más detalles sobre las fuentes de restablecimiento de AVR®, visite Descripción general de fuentes de restablecimiento de AVR.

Materiales

Herramientas de hardware



ATmega328PB Xplained Mini

Kit de evaluación

Herramientas de software



Estudio Atmel®

Entorno de desarrollo integrado

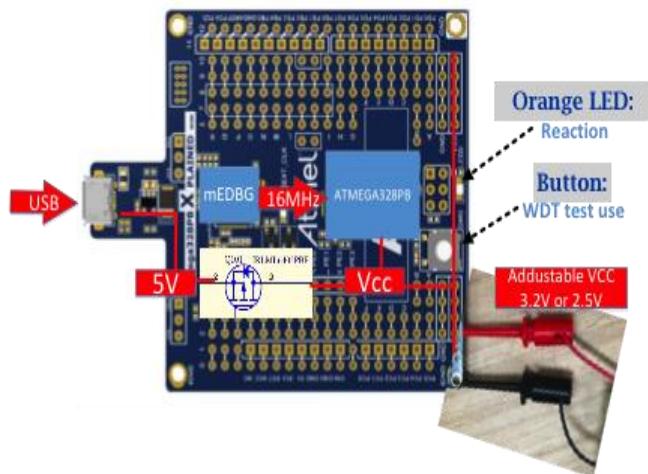
Archivos de ejercicio

<https://microchipdeveloper.com/install:example-and-exercise-files>

Archivos adicionales

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42469-ATmega328PB-Xplained-Mini_User-Guide.pdf

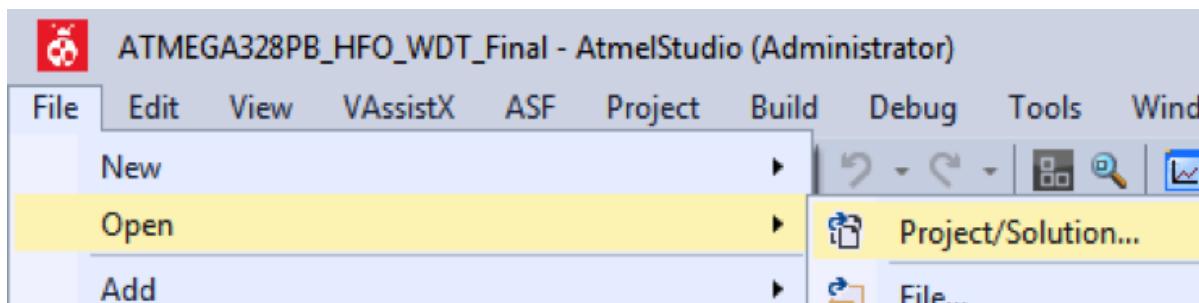
Diagrama de conexión



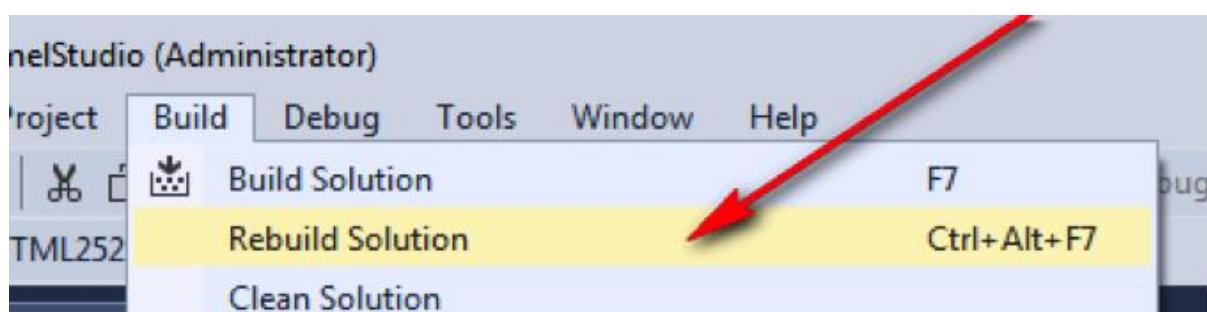
Procedimiento

1 Tarea 1 - Abrir y compilar proyectos

- Descargue los archivos de origen del proyecto de la sección Archivos de ejercicio anterior y descomprimalos en su computadora.
 - Abrir Atmel Studio 7
 - Seleccione archivo > abrir > proyecto/solución



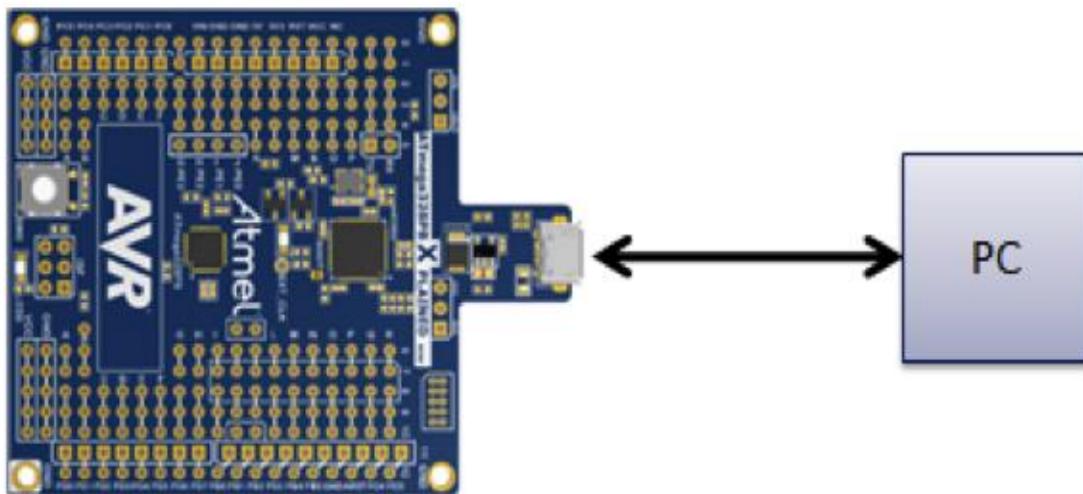
- Seleccione ATMEGA328PB_HFO_WDT_Final.atsln de los archivos de proyecto descargados.
 - En el menú Generar, seleccione 'Reconstruir solución'.



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

2 Tarea 2 - Preparación de la Junta

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

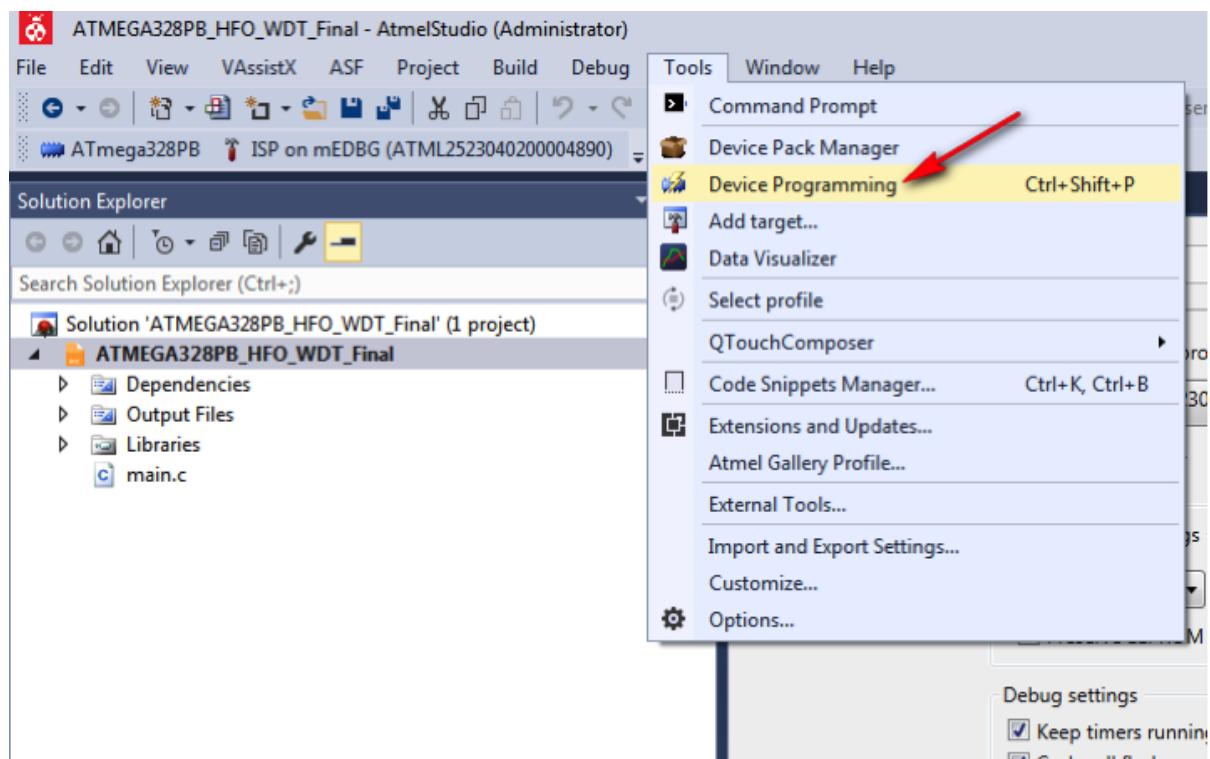


3 Tarea 3 - Configuración del programador

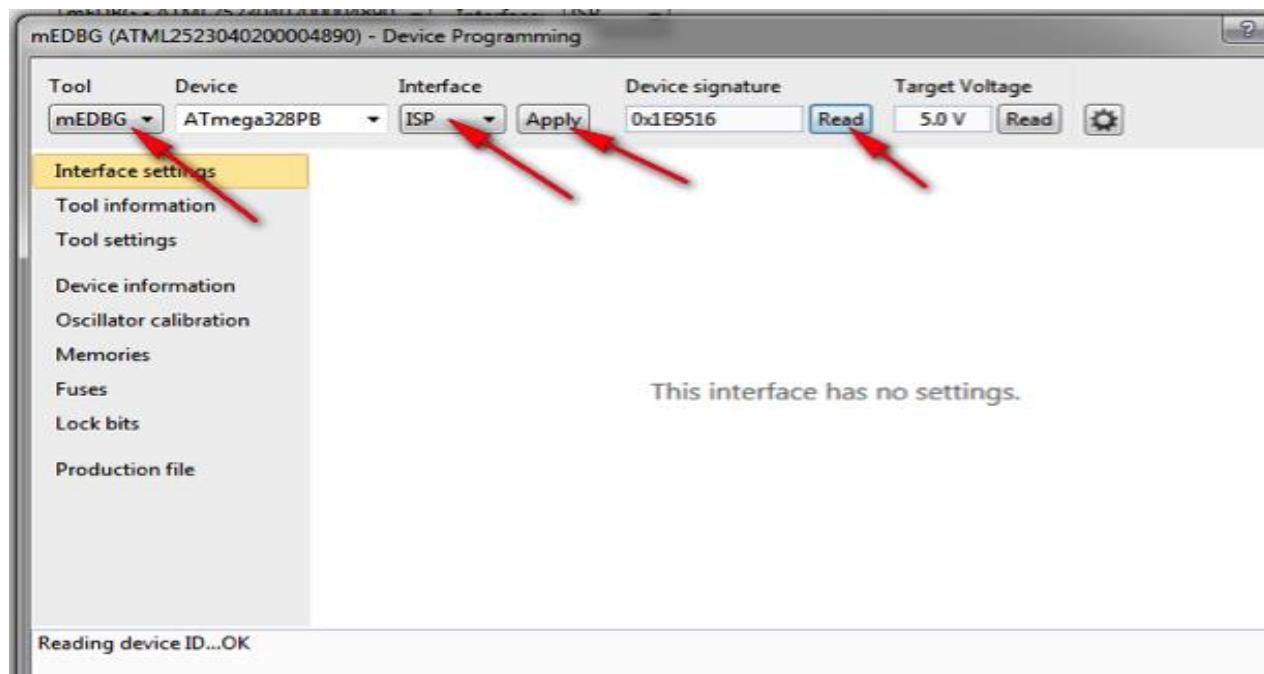
Si debugWire está habilitado, desactívelo.

Deshabilitar debugWire (DWEN) ▶

- Haga clic en Herramientas > programación de dispositivos:

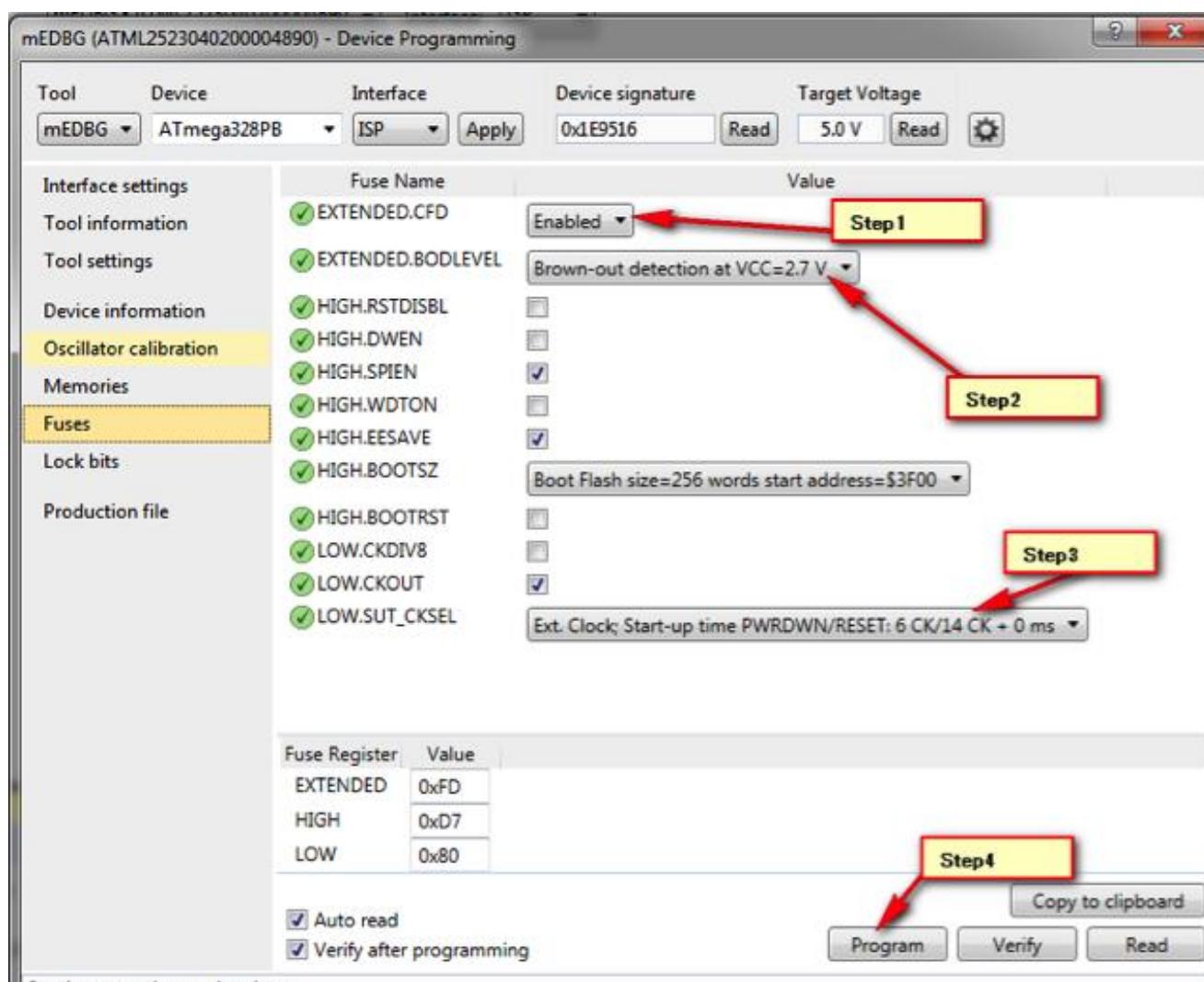


- Haga clic en los botones que están marcados como flechas rojas en la captura de pantalla.



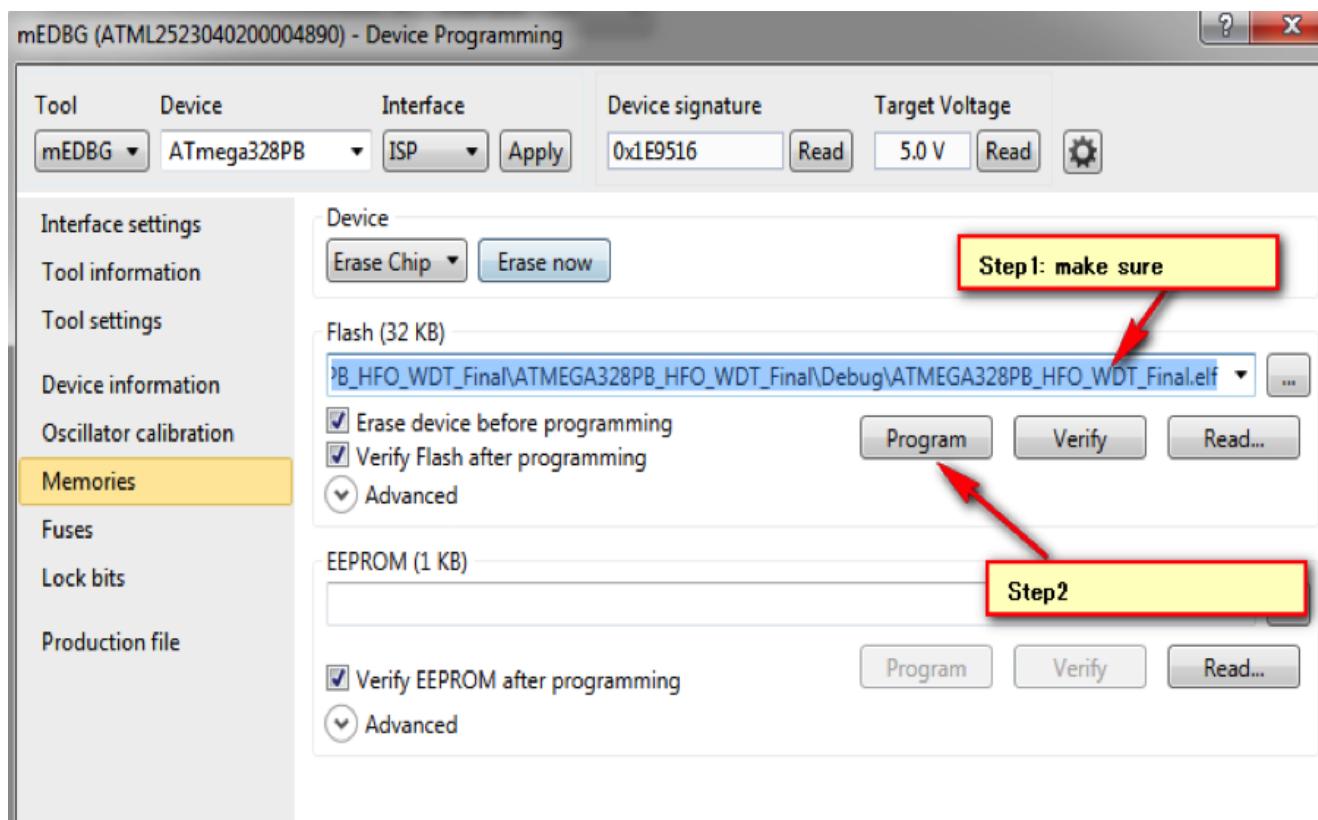
4 Tarea 4 - Programación de los bits de fusible

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



5 Tarea 5 - Programación de dispositivos

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



6 Tarea 6 - Código del proyecto

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

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

```
#define F_CPU 8000000UL
```

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

```

#include <util/delay.h>

//initialize watchdog

void WDT_Init(void)

{

//disable interrupts

cli();

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

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

(1<<WDCE)|(1<<WDE )|                         // Enable Watchdog System Reset Mode if unintentional

ly enabled.

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

//Enable global interrupts

sei();

}

//Watchdog timeout ISR

ISR(WDT_vect)

{

//Burst of 0.1Hz pulses

for (uint8_t i=0;i<4;i++)

{

//LED OFF

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

_delay_ms(80);

//LED ON

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

_delay_ms(20);

}

```

```

}

#define BORFbit 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

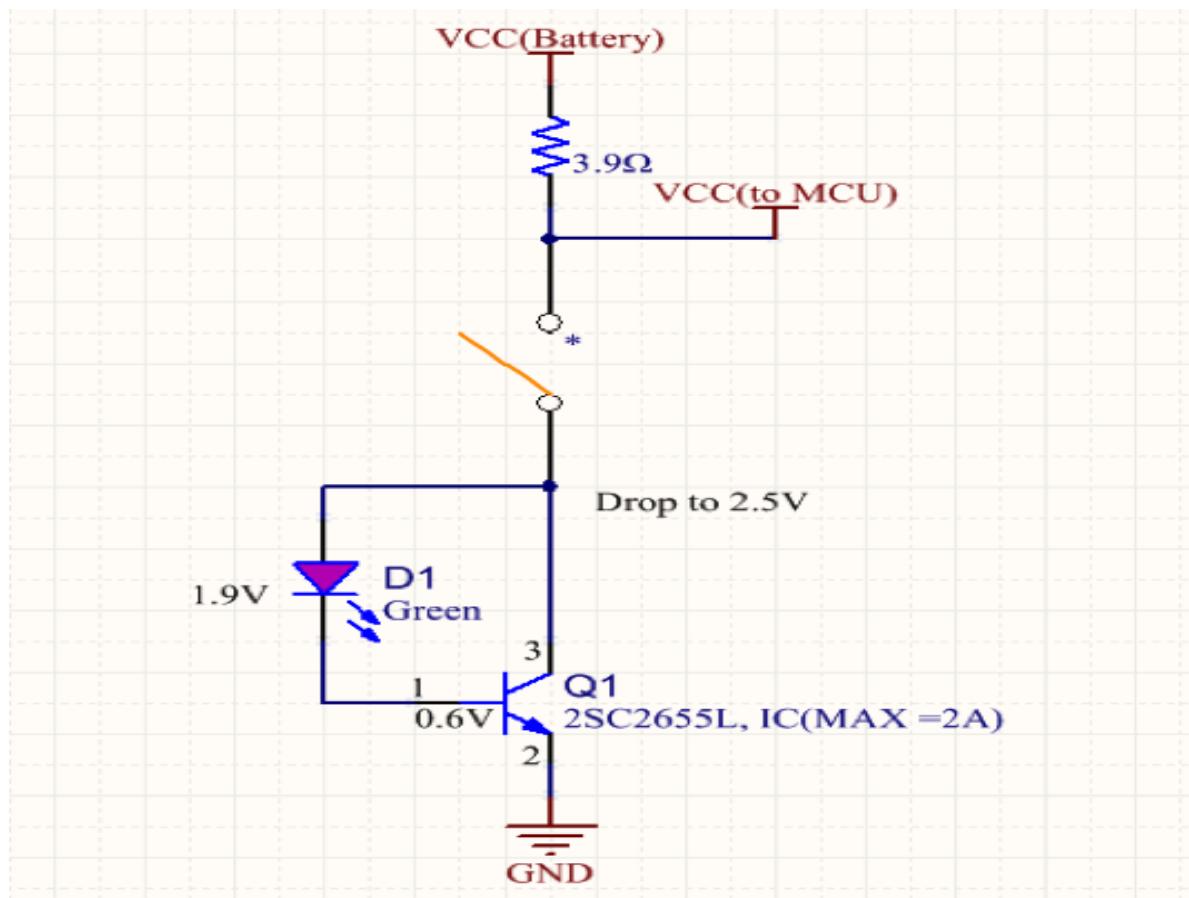
```



```
    wdt_reset();  
}  
  
}
```

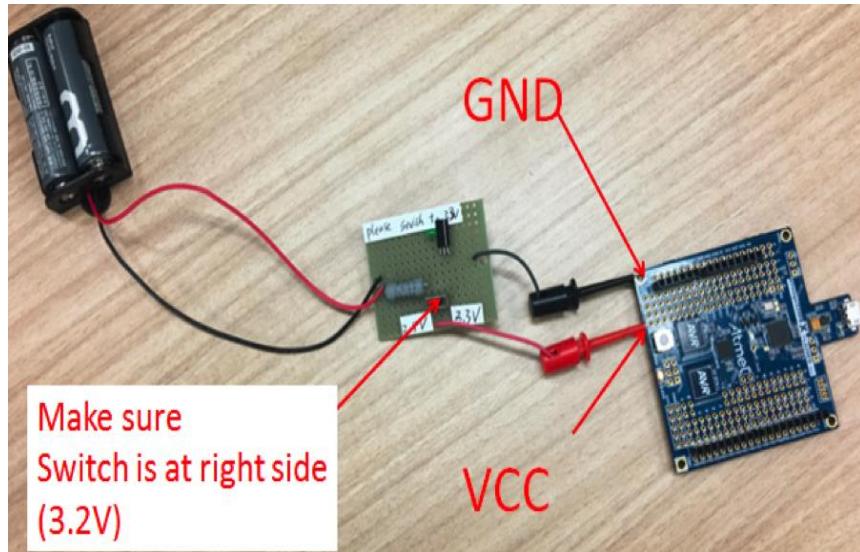
7 Tarea 7 - Restablecer circuito de prueba

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



8 Tarea 8 - Prueba POR

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



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

```

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

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

```

9 Tarea 9 - Prueba BOR

- 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) en función del código resaltado a continuación porque se detecta el restablecimiento de BOR.

```

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

```

10 Tarea 10 - Prueba WDT

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

Nota: El temporizador WDT está configurado en 2 segundos.

```

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

```

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

```

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

```

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

```

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

```

Análisis

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

Conclusiones

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