GUÍA ATMEL STUDIO 7

Desarrollado por Raúl Alzate & David Sterling en agosto - 2018 para el curso de B-Learning de señales y sistemas.

# Introducción a Atmel Studio:

Atmel Studio es un **IDE** desarrollado por la compañía Atmel Corporation. Este software fue creado para la programación de todos los productos de microcontroladores de esta empresa. Hasta la fecha la versión 7 es la última desarrollada antes de que Atmel fuera comprada por Microchip, también fabricante de microcontroladores.

Al igual que los demás IDE’s, Atmel Studio 7 facilita el desarrollo de software para microcontroladores ya que realiza **highlight** de las palabras claves, realiza funciones de **autocompletar** y además **indica errores de sintaxis** antes de realizar la compilación.

Sin embargo, ofrece otras características adicionales muy importantes para el desarrollo de microcontroladores como son las funciones de **depuración** (**debug on chip**), **simulación**, entre otros.Estos últimos temas se explorarán mas adelante en esta mima guía.

## Instalación de Atmel Studio 7

El desarrollo para microcontroladores involucra en primer lugar un compilador apto para la arquitectura de microcontroladores que se desea trabajar. En segundo lugar, se debe disponer de una herramienta hardware que permite la programación de estos dispositivos. En el caso de Atmel los protocolos mas populares para esta labor son JTAG y SPI. Es muy común en las tarjetas de desarrollo que el hardware de programación este junto con el microcontrolador en la misma board. De esta manera se simplifica el trabajo del desarrollador ya que basta con hacer una conexión mediante USB para poder programarlo. Sin embargo, no siempre es el caso y por eso es muy importante tener presente que siempre se necesita disponer de un hardware para llevar a cabo la programación de los microcontroladores.

Este hardware de programación implica a su vez la necesidad de unos drivers, los cuales también son obligatorios en el proceso de instalación. Finalmente, el IDE propiamente hablando (Atmel Studio 7), el cual facilitara la integración de los elementos anteriormente citados.

Cuando se desea desplegar una aplicación para microcontroladores, el IDE en primer lugar debe invocar al compilador el cual convertirá el código fuente en lenguaje de máquina.

# Creación de Proyectos en Atmel Studio 7

Se pueden crear 4 tipos diferentes. Estas opciones incluyen la creación de un proyecto usando lenguaje ensamblador, proyecto de C/C++ para librería o para aplicación y finalmente proyecto con una tarjeta de desarrollo.

## Proyectos En Ensamblador

## Proyecto Para Creación De Librería C/C++

Los proyectos que son tipo librería son muy similares los proyectos para aplicación con la diferencia que no se genera un ejecutable. También es muy importante tener en cuenta que si la librería se involucra directamente con los periféricos del microcontrolador es muy posible que la librería pierda generalidad. Lo anterior se debe a que los periféricos disponibles pueden variar de acuerdo a la referencia de los microcontroladores e incluso los nombres de los registros pueden variar. Esto se vera mas adelante en los proyectos de ejemplo.

## Proyecto Para Creación De Aplicación C/C++

## Proyecto Para Trabajar Con Boards Basadas En Microcontroladores De Atmel

Cuando se trabaja directamente con microcontroladores de Atmel, es muy común el uso de programadores. Sin embargo, en el caso de tarjetas como Arduino, que se basan en microcontroladores de Atmel, se suele hacer uso de chips adicionales en la misma tarjeta para realizar comunicación serial entre el microcontrolador y la maquina de desarrollo y también para realizar la programación del microcontrolador. Lo anterior tiene la ventaja de permitir un manejo muy compacto del hardware ya que basta con conectarlo por usb a la máquina de desarrollo. Para esta clase de hardware se debe seleccionar la opción ASF BoardProject o Project From Arduino Sketch.

# Programación De Los Microcontroladores

Para llevar a cabo la programación de estos dispositivos normalmente se requiere de un protocolo de comunicaciones entre el programador y el microcontrolador. El protocolo más común es el JTAG sin embargo también se dispone del protocolo SPI. El SPI tiene la ventaja de que requiere menos pines del microcontrolador, pero a su vez no es apto para realizar depuración, lo cual se discutirá más adelante en esta guía. Por el contrario, el protocolo JTAG permite hacer programación y depuración.

En cualquier caso, cuando se usa Atmel Studio, lo primero es conectar los pines del microcontrolador al programador. Posteriormente se accede a la pestaña de herramientas (tools) y se selecciona la opción Device Programming como se muestra en la Ilustración 1.

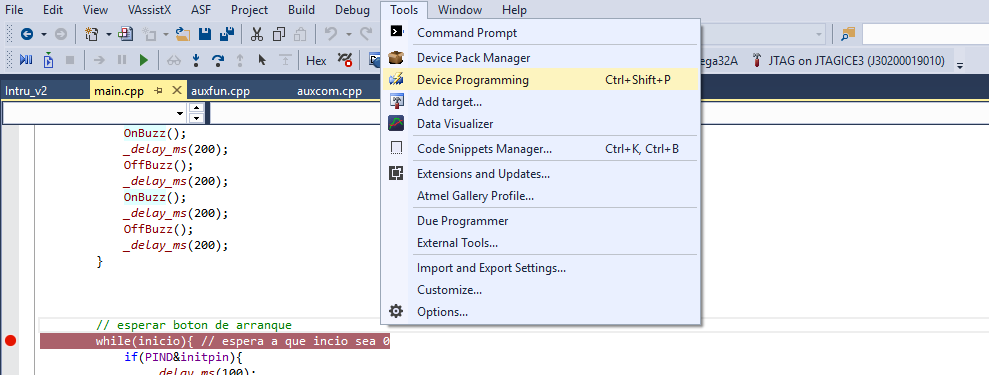


Ilustración 1. Apertura de la ventana de programación de microcontroladores.

## Conexión Hardware Para La Programación

La forma de conexión depende del protocolo que se va a emplear para hacer la programación de los microcontroladores. En este guía vamos a considerar los protocolos principales: JTAG y SPI pero se debe tener en cuenta que hay muchos más. El protocolo empleado también trae limitaciones a la velocidad máxima posible de programación, lo cual seria un factor importante cuando se desea producir boards en serie.

JTAG:

En la figura Ilustración 1 se muestran los pines que se usan para hacer la programacion con JTAG de los microcontroladores.

Si se compara con la Ilustración 2, que corresponde al caso con programacion SPI, se puede apreciar que el JTAG requiere mayor cantidad de pines para hacer la programación. Sin embargo, la gran ventaja de este protocolo es que también se puede usar para depuración.

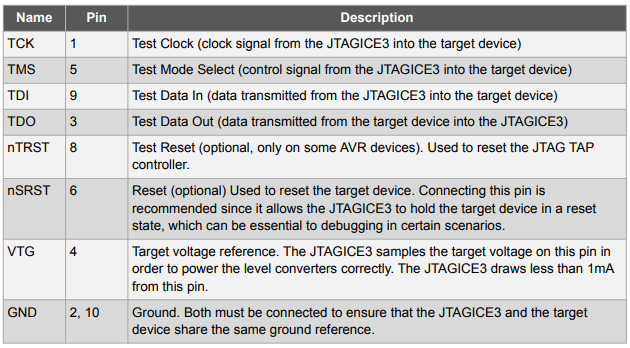


Ilustración 1. Tabla de pines para conexión con JTAG usando el JTAGICE3

SPI: En la Ilustración 2 se muestran los pines del JTAGIC3 que se usan para realizar la programacion. Al igual que ocurre cuando se va a conectar dos circuitos entre sí, es necesario que compartan la tierra. GND corresponde a la tierra del circuito del microcontrolador. VTG hace referencia a la tensión de alimentación del microcontrolador. /Reset corresponde al pin de reset del microcontrolador, el cual es necesario que pueda ser controlador por el programador. Lo anterior se debe a que el programador puede usar este reset para asegurarse que el CPU este detenido. SCK corresponde a la señal de reloj que se usa para sincronizar la comunicación entre el microcontrolador y el programador. MISO y MOSI son los pines que se usan para el envío y recepción de datos. El protocolo JTAG se basa en el modelo maestro-esclavo. MISO significa Master Input Slave Output mientras que MOSI es Master Output Slave Input.

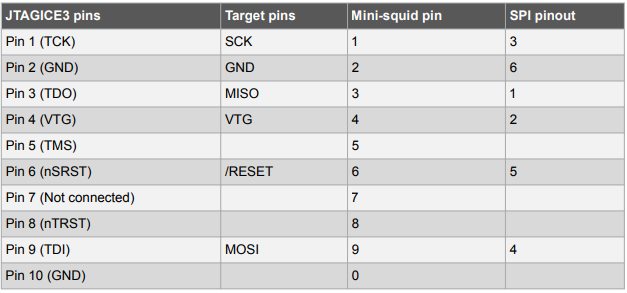


Ilustración 2. Tabla de pines para programación con SPI usando el JTAGICE3.

En la Ilustración 2 se muestran las funciones de cada uno de los pines del microcontrolador ATtiny4313. Se puede observar que la mayoria de los pines desempeñan mas de una funcion a excepcion de los pines de alimentacion VCC y GND. Los pines necesarios para hacer la programacion del microcontrolador con el protocolo SPI son: PA2, GND, PB5, PB6, PB7 y VCC. Se puede observar que estos pines ademas de servir para la programacion del microcontrolador tambien pueden ser usados como salidas o entradas digitales

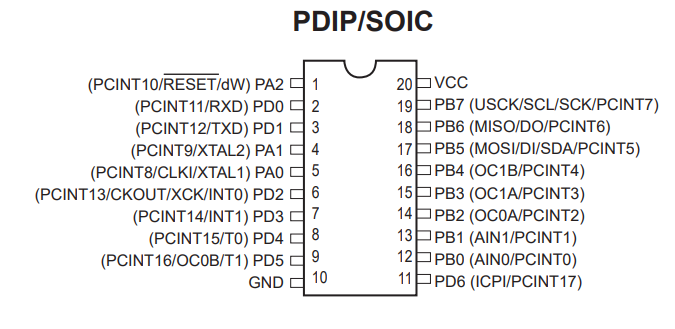


Ilustración 3. PinOut del microcontrolador AVR-8 bits ATtiny4313.

En la Ilustración 3 se muestra la conexión de la alimentación y un cristal de cuarzo a un microcontrolador ATtiny4313. En la Ilustración 2 se puede ver que los pines donde se debe conectar el cristal son los que están etiquetados con XTAL, es decir los pines PA0 y PA1. Para llevar a cabo la programación de un microcontrolador de la familia AVR es necesario que estos dispongan de una señal de reloj. Normalmente los microcontroladores de esta familia están configurados por defecto para usar un oscilador de cristal, sin embargo, esta configuración se puede cambiar. Esto se explicará más adelante en la sección de clock fuses.

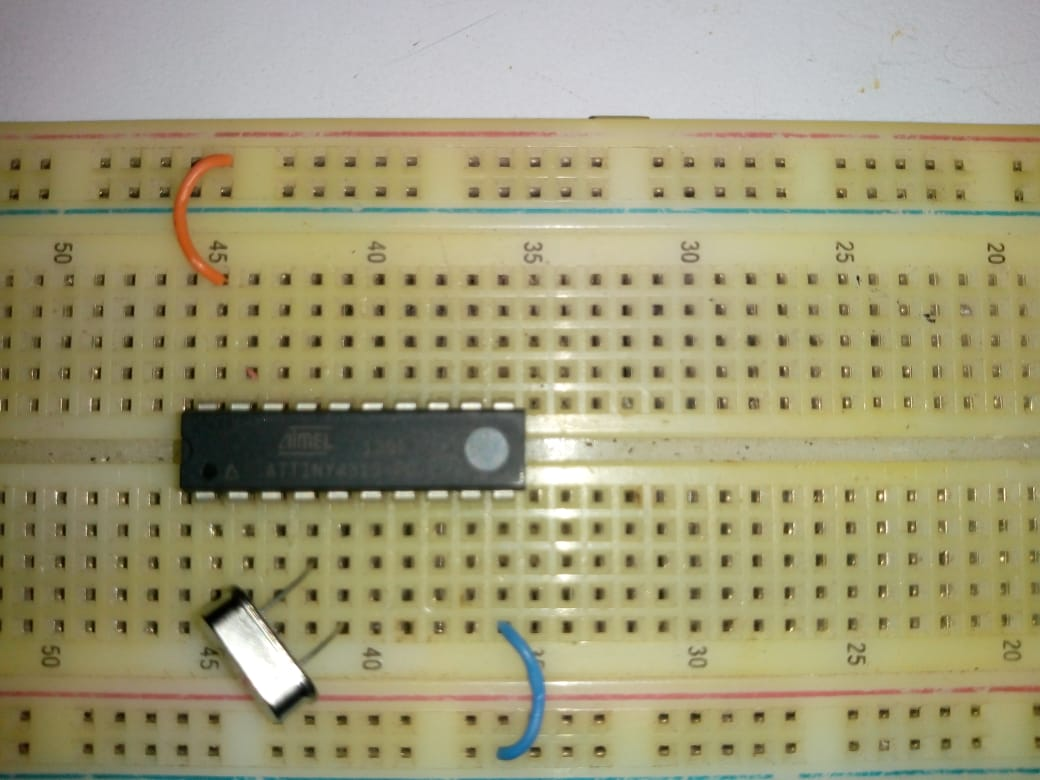


Ilustración 4. Conexión de la alimentación y el cristal de cuarzo para el microcontrolador ATtiny4313.

En la Ilustración 6 se muestra la conexión del programador al microcontrolador para poder programarlo. Se puede observar que se agrega una resistencia de pull-up entre Vcc (alimentación positiva) y el pin de reset. El esquema eléctrico se muestra en la Ilustración 4.

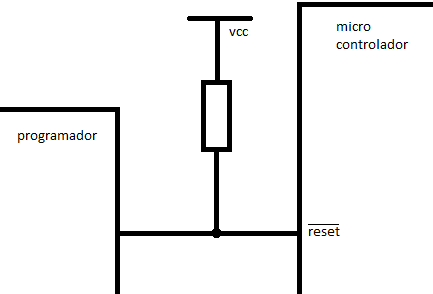


Ilustración 5. Conexión de la resistencia de Pull-Up para el pin de reset.

Para hacer un reset del microcontrolador se debe establecer un nivel lógico bajo (GND). Cuando se desea operación normal del microcontrolador se debe poner un nivel lógico alto en el pin de reset. Y que el programador debe hacer un hacer uso de este pin, será necesario imponer en el mismo niveles altos y bajos durante la fase de programación. Por lo anterior se usa la resistencia de pull-up. En la figura Ilustración 5 se puede ver que el programador posee 3 leds, pero solo el de la mitad esta encendido. Este led indica que el programador esta conectado a una fuente de alimentación.

|  |  |
| --- | --- |
|  |  |

Ilustración 6. Conexión del programador JTAGICE3 usando el protocolo JTAG.

En la Ilustración 6 se muestra cómo se enciende el led verde del programador cuando se hace tiene un valor correcto de alimentación para el chip.

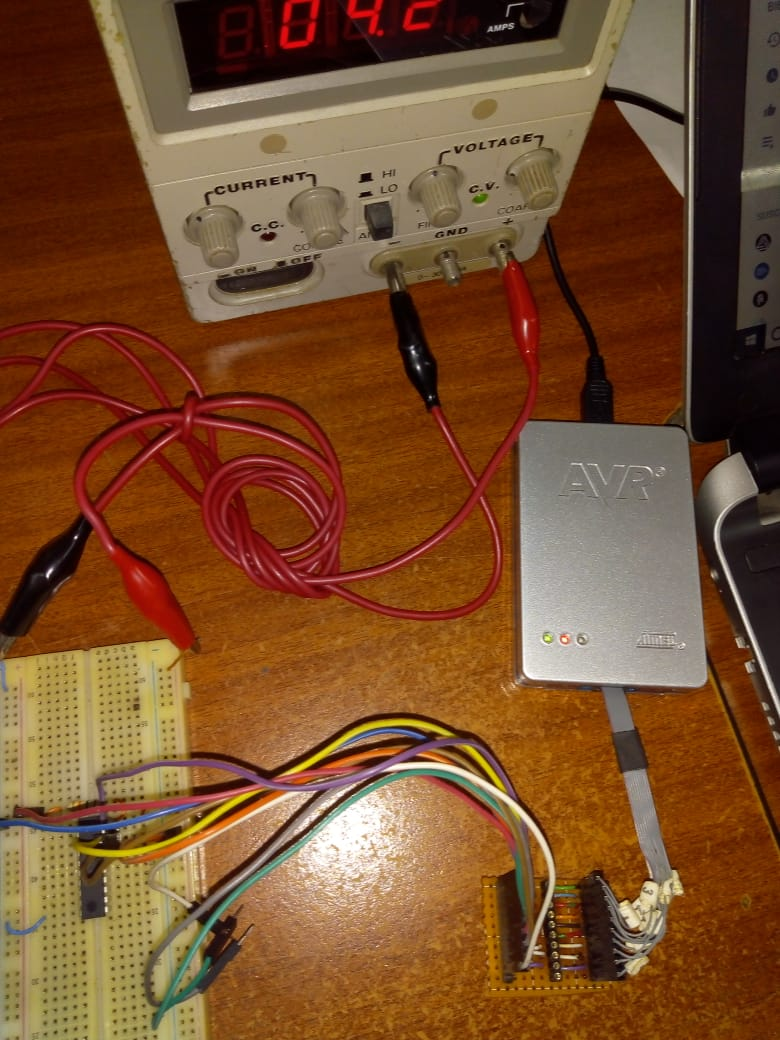


Ilustración 7. Programador y microcontrolador correctamente conectados para llevar a cabo la programación.

Una vez se han hecho las conexiones necesarias, se procede a ejecutar atmel studio y abrir la ventana de programacion de dispositivos (Ctrl+shift+p). En la Ilustración 7 se muestra en la esquina superior izuierda que en la opcion tool se seleccióna el programador, en este caso el JTAGICE3. Ademas se debe seleccionar la referencia del microcontrolador que es ATtiny4313. Este microcontrolador solo soporta programacion por SPI. Sin embargo la mayoria de las referencias de AVR que no son de la familia Tiny soportan JTAG y SPI. Tambien se puede observar en la misma figura que se selecciona SPI como interfaz de programacion. Una vez se selecciona la interfaz de programacion se debe dar click en el boton de aplicar (apply). Seguido de esto, donde dice device signature se debe dar click en read. Si la conexión esta bien hecha y el microcontrolador tiene el cristal de la frecuencia correcta entonces se mostrara un numero en hexadecimal en la casilla device signaure. Esto indicara que el disposittivo esta listo para ser programado.

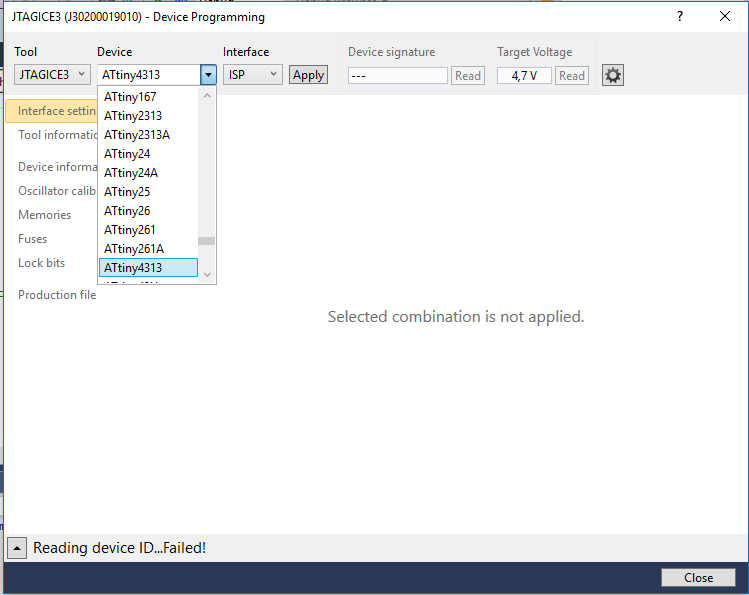


Ilustración 8. Selección del dispositivo a programar.

## Programación de Fuses

La mayoría de las funcionalidades del microcontrolador se configuran estableciendo valores en los registros del mismo. Sin embargo, en el caso de Atmel hay algunas características que se configuran en unas memorias separadas a las que llaman “fuses” (de ahora en adelante fusibles). Estas regiones de memoria no son accesibles durante ejecución, por consiguiente, no es posible modificarlos mediante instrucciones de máquina.

Entre las características más importantes que se configuran con los fuses están:

* Fuente de reloj para CPU (CPU Clock source)
* Frecuencia de reloj (CPU Clock)
* Brown-Out Detector
* Watch Dog Timer
* Modo Debug Wire
* Memory lock
* Entre otros.

La fuente de reloj puede ser generada usando un cristal de cuarzo o puede generarse internamente en el microcontrolador con un oscilador RC (que no tan preciso como el oscilador de cuarzo). También es posible establecer que la frecuencia de reloj no se genere, sino que esta se recibirá externamente.

El Brown-Out detector es un hardware del microcontrolador que se encarga de detectar cuando ocurre una reducción de la alimentación del microcontrolador. Cuando esto ocurre es posible que algunos de los registros del microcontrolador experimenten cambios deseados en sus bits lo cual puede ser peligroso dependiendo de la aplicación que se esté trabajando. Por este motivo cuando ocurre un bajón de tensión del microcontrolador, lo mas seguro es realizar un reset. Cuando el fusible de Brown-Out Detection está habilitado, el microcontrolador hace un reset automático si ocurre una reducción de la tensión por debajo de un nivel especificado.

El Watch-Dog-Timer es un temporizador que realiza un reset del microncontrolador cuando este se desborda. Esta característica es muy importante cuando se está implementado una aplicación que hace uso de operaciones matemáticas muy complejas. En este caso puede ocurrir que en medio de los cálculos matemáticos provoquen que el programa se quede enganchado en un ciclo infinito, y puede ser preferible hacer un reset. Cuando se hace uso del Watch-Dog-Timer siempre se debe hacer un clear del temporizador al finalizar cada ciclo del programa.

Otra característica muy importante a nivel de producción es el Memory-Lock. Cuando se activa este fusible ya no es posible leer ni escribir sobre la memoria de programa y los fuses. Esto puede ser muy útil para evitar que otras empresas realicen espionaje industrial del software programado en los microcontroladores. Sin embargo, si esto se hace en fase de pruebas sería equivalente a perder el microcontrolador pues ya no se podrá reprogramar para hacer ninguna clase de mejoras.

## Escribiendo la Memoria De Programa

Lo primero que se debe hacer es compilar la aplicación sin errores y preferiblemente sin advertencias (warnings). La compilación se puede lograr con el shortcut Ctrl + p. Seguido de esto se debe abrir la ventana de device programming. En esta ventana se selecciona el programador que se va usar, el cual debe estar debidamente conectado al microcontrolador de acuerdo al protocolo de comunicación seleccionado. Después se debe seleccionar el dispositivo que se desea programar de la lista, tal como se muestra en la figura tal.

# Proyecto de Ejemplo

Para poner en practica los conceptos vistos hasta el momento se llevará a cabo el parpadeo de un led usando los 4 tipos de proyectos presentados hasta el momento. Este tipo de proyectos se les suele dar el nombre: “hola mundo de la electrónica”, ya que corresponde al ejemplo más sencillo para el caso hardware.

Mas adelante en esta guía se presentan ejercicios más avanzados.

## Parpadeo De Un Led En Ensamblador

Para hacer parpadear un led lo más sencillo es hacer uso de las salidas digitales del microcontrolador. La estructura de los registros puede cambiar de acuerdo al fabricante. En este caso se va a tomar como ejemplo el caso de Atmel. Para con las entradas y salidas digitales se usan los siguientes registros: DDRx, PINx y PORTx. Normalmente las salidas digitales se agrupan de acuerdo al tamaño de los registros. En el caso de las arquitecturas de 8 bits, se agruparían 8 pines en un puerto. En el caso de Atmel estos grupos de pines se designan con una letra, empezando desde la A. Si una referencia particular de microcontrolador tiene puertos A y B entonces se dispone de un registro DDRA y un registro DDRB. Para estos registros

# Solución de Problemas y Depuración

Cuando se presentan errores que no corresponden problemas de sintaxis, estos no son señalados por el IDE. En muchas ocasiones esta clase de errores corresponden a errores en la lógica del diseñador y de acuerdo a la envergadura del proyecto puede ser muy difícil detectarlos en el código mediante simple inspección.

Por este motivo es muy común llevar a cabo “Debugging” que en español se traduce como depuración. En el caso de programas que se ejecutan en un computador de escritorio, la depuración consiste principalmente en un modo especial de ejecutar un programa bajo desarrollo. En este modo de ejecución, cada instrucción se ejecuta una por una según el programador lo ordena mediante la interfaz grafica de usuario. Al mismo tiempo, se puede ver mediante con ayuda del IDE los valores que toman las variables a medida que se van ejecutando las instrucciones. De esta forma un programador puede hallar de forma más fácil los errores en su programa.

En el caso de desarrollo con microcontroladores es muy similar, pero hay unos nuevos conceptos adicionales que hay que tener en cuenta. En primer lugar, esta clase de programas no corren en la misma máquina de desarrollo (Equipo en el que se está escribiendo el programa), sino que corren en el microcontrolador, el cual es ajeno al sistema operativo en el que se esta desarrollado y como consecuencia el IDE por si mismo no puede hacer depuración. Lo anterior significa que es necesario disponer de un hardware adicional que le permita a la máquina de desarrollo controlar la ejecución de las instrucciones en el microcontrolador y además poder ver valores de variables y registros. A este hardware se le llama Debugger en inglés, y por todo lo anterior se le suele dar el nombre de on-chip Debugging cuando se trata de depuración con microcontroladores.

En algunas ocasiones los fabricantes integran la función de depuración al programador, pero también es común que se requieran productos separados para cada labor, un producto para programación y otro para depuración. Lo anterior ocurre normalmente cuando las funcionalidades de depuración son muy avanzadas y por consiguiente el hardware para implementarlas se hace complejo.

Ejemplos de programadores y Debuggers son: JTAG ICE3, PicKit4, ST-LINK V2, Cold Fire, entre otros.

Para realizar la depuración normalmente se hace uso de un protocolo de comunicación entre el microcontrolador y el Debugger. Es muy común el uso de protocolos como JTAG, el cual, a su vez, se usa también para programar el microcontrolador.

# Uso del simulador con Atmel Studio 7

Uno de los primeros pasos para validar el programa diseñado para un microcontrolador dado es realizar una simulación. Esto puede servir para verificar que los comandos alteren los registros correctos del microcontrolador bajo ciertas condiciones de operación.

Para hacer un primer acercamiento a la herramienta se va a analizar a continuación el ejemplo mas sencillo que es parpadear un led. En la Ilustración 9 se muestra resaltada la

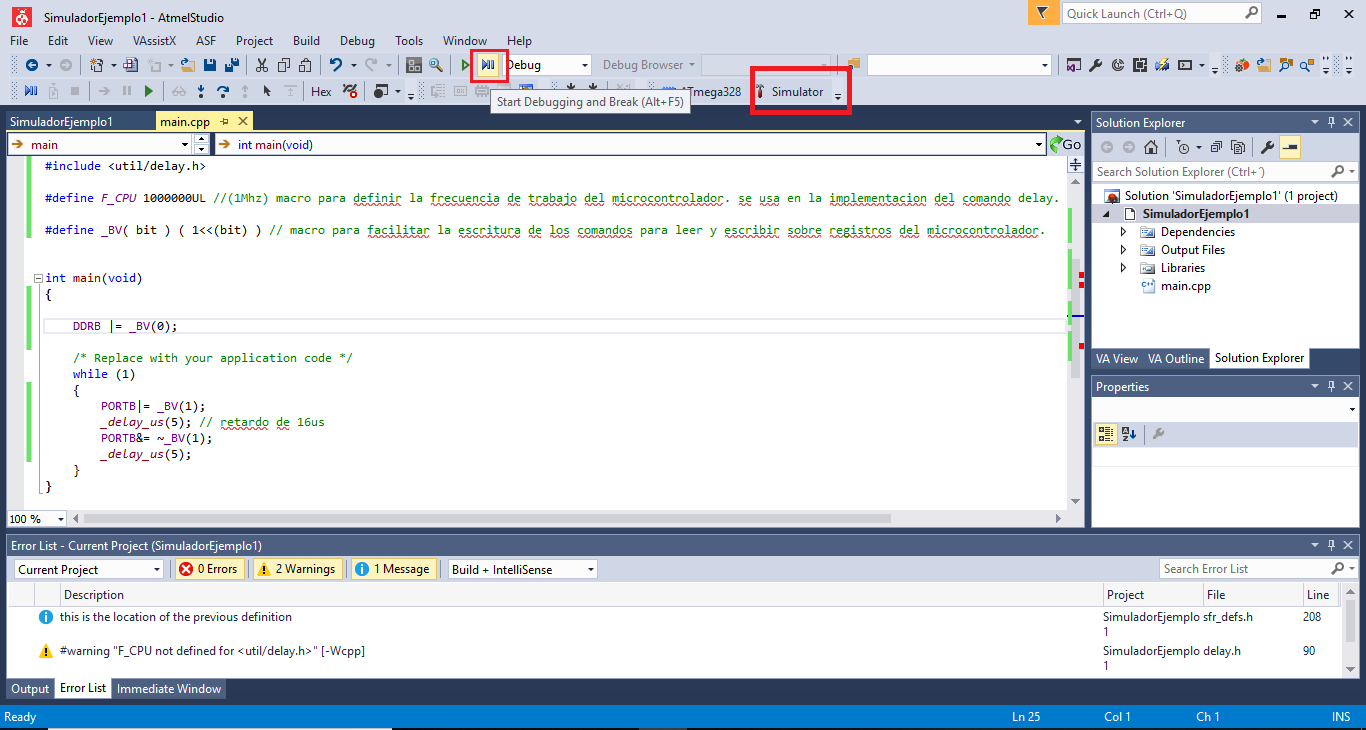


Ilustración . Ejemplo para el uso del simulador con Atmel Studio 7.

# Proyectos De Ejemplo Atmel Studio 7

## Uso de Temporizadores

Los temporizadores son circuitos digitales que permiten contar eventos. Cuando esos eventos son el flanco de subida de una señal cuadrada periódica entonces el temporizador permite medir el tiempo transcurrido. Es muy importante distinguir que a pesar de que un temporizador se encuentre en el mismo chip del CPU del microcontrolador, se trata de entidades diferentes. Precisamente esto es lo que les da valor a los temporizadores, ya que, al tratarse de hardware independiente a la CPU, permite contar eventos sin llevar a cabo ninguna instrucción en el CPU. De manera muy similar ocurre con los demás periféricos de los microcontroladores, se introducen como entidades separadas dentro del mismo chip para permitir

## Uso de Convertidores Analógico Digital (ADC)

Para hacer uso del convertidor analógico digital del microcontrolador basta con establecer los valores apropiados en los registros del ADC. Para esto es muy importante tener en cuenta los fundamentos básicos del proceso de conversión Analógico digital. Para este caso vamos a tomar de ejemplo el microcontrolador atmega328, el cual ha sido muy popular gracias al Arduino 1. Antes de empezar con los registros es necesario realizar una conexión básica en el hardware.

Después de las conexiones básicas de alimentación y resistencia de pull-up para el pin de reset, es necesario que el ADC disponga de una tensión de referencia.

## Uso de Convertidores Digital a Analógico (DAC)

Para el caso de los convertidores Digital Analógico, el manejo es muy similar al del ADC si el convertidor está incluido como periférico en el microcontrolador. Sin embargo, si el microcontrolador no tiene disponible un DAC, es posible usar una señal del tipo PWM en combinación con un filtro pasa-bajo para lograr este tipo de conversión.

## Uso de Comparadores

## Implementación De Un Controlador PID

# Referencias

Datasheet jtag

[1] Datasheet attinny4313

Datasheet atmega328