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

◎ Objetivo

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

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

☑ Materiales

Herramientas de hardware (*opcional*)

Herramienta		❷Sobre	
(http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	Mini kit de evaluación **ATmega328PB Xplained**	❷ (http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	🛒 (https://www.microchipdirect.com/prc

Herramientas de software

Herramienta	❷Sobre	Instaladores			Instrucciones de 👣 instalación
		⊞ventanas	△ Linux	🍎 Mac OSX	
Atmel® Studio ▣ Integrated Development Environment	🗗 (/atstudio:start)	⭳ (http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe)	⭳	⭳	🗗 (/install:atstudio)

Exercise Files

| File	⊞ Windows	
▤ **Main.c Source File**	⭳ (https://microchiptechnology.sharepoint.com/:f:/s/DeveloperHelp/EgvbV6I04yBAvXfI6BemPdoBdqWAfKoruX9pkt21RgFG-w?e=pCb004)	⭳ (ht

Additional Files

Files

📄 **Xplained Board User Guide** (http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf)
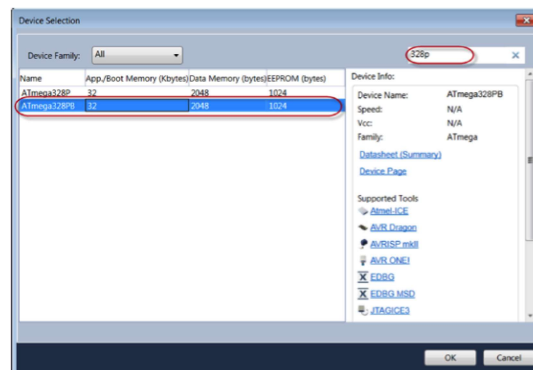
👣 Procedure

1   **Task 1 - Project Creation**

- Open Atmel Studio 7
- Select **File > New > Project**
- Select GCC C Executable Project and give it the name **Project1**
- Choose a location to save the project on your computer
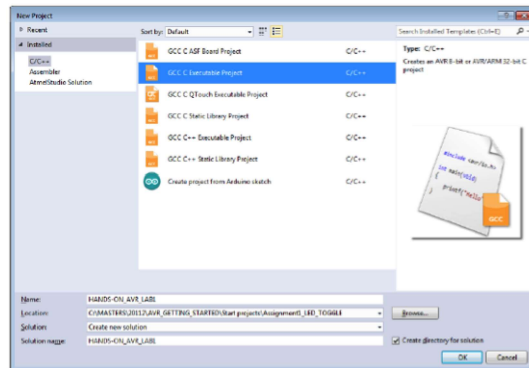
(/local--files/8avr:avradc/step1.png)

- The Device Selection window will appear. In the search bar enter **328P**, then select the device **Atmega328PB** and click OK.

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

2   **Task 2 - Main.c**

This project reads the internal temperature sensor, converts the result to degrees centigrade, then stores the result in **ADC Temperature Result**.

1) The `main.c` file is where the application code is added. The project has a `main.c` file already created but it only contains a `while(1)` statement. Modify `main.c` by entering the lines in the gray code block below.

⚠️ #include `<avr/io.h>` is automatically added to the `main.c` file when it is generated. This should always be placed before the `main(void)` loop. The header file `io.h` calls the `iom328pb.h` file that defines the ADC register definitions.

```c
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;

}
```
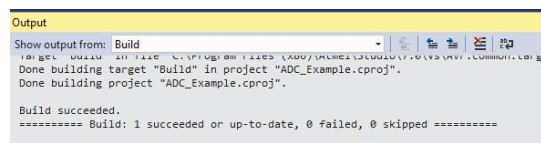
3   Task 3 - Build Project

- Select **Build > Build Solution** from the Studio 7 menu to compile the code. You will see a **Build Succeeded** message in the output window. If there are any errors, check `main.c` for any mistakes in entering the program code.
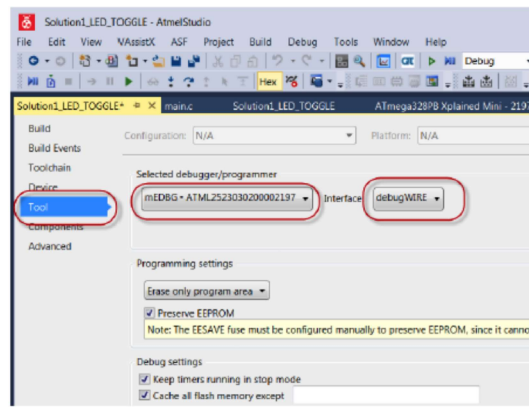

(/local--files/8avr:avradc/output.png)

4   Task 4 - Programming the Xplained Board

- Connect the Xplained board to the USB port of the computer using the included cable.
- In the **Solution Explorer** area, right-click on the project name and select **Properties**.
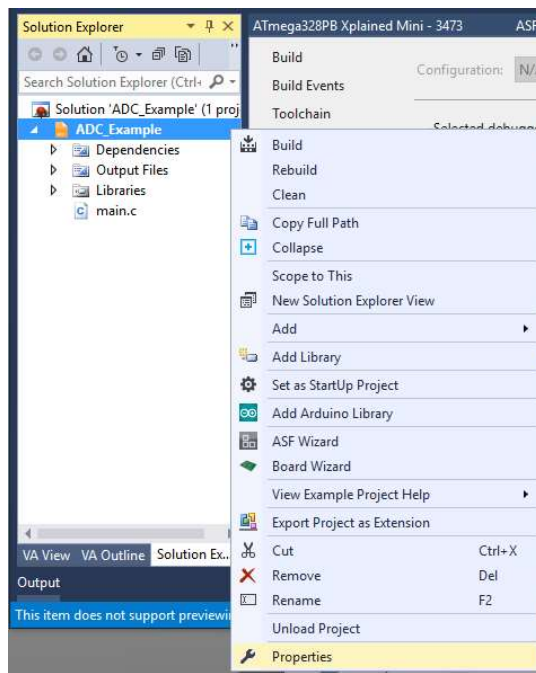
(/local--files/8avr:avradc/solution.png)

- Under the **Tool** menu selection, choose the **mEDBG** and **debugWire** as the interface.

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

- Select **Debug > Start Without Debugging** from the Studio 7 menu. The project will build and then program the xplained board with the project code along with debug control.

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

- Studio 7 will show a **Ready** message when the programming is complete.

⚠️ If the Xplained board will not connect, there may be a fuse setting causing this to occur (/boards:debugbrick).

5   Task 5 - Debugging

Debugging a device is essential for determining how a program may be running.

- Select **Debug > Start Debugging and Break.** The project will build and the program will be loaded into the Xplained board
- The I/O View window will open up showing the various peripherals
- Click on the **ADC** selection to open the I/O view for the ADC
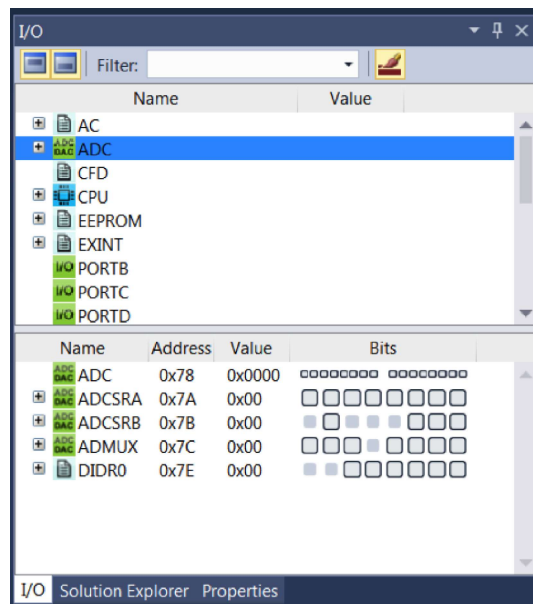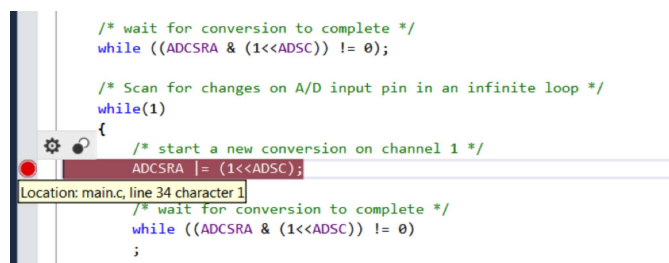
(/local--files/8avr:avradc/adcdebug.png)

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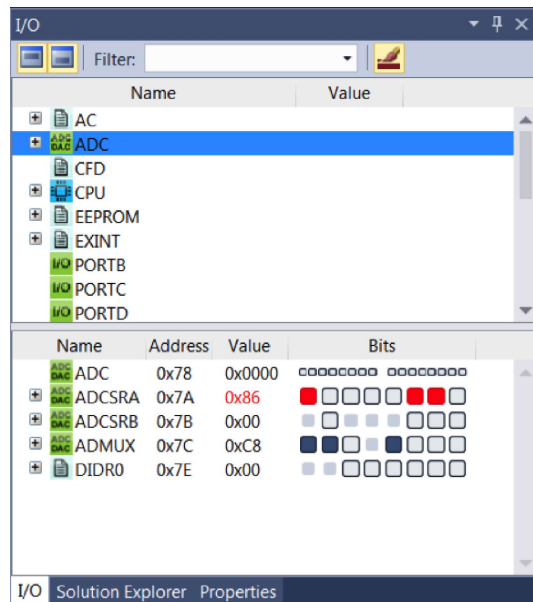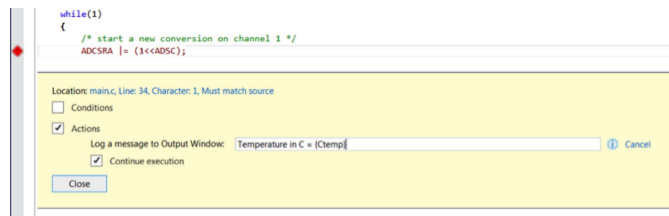

(/local--files/8avr:avradc/adcdebug2.png)

6   Task 6 – Breakpoint and Output

- Click on the **Debug > Break All** from the top menu of Studio 7
- Click on the margin to enable a breakpoint on the command line at the ADCSRA statement within the While loop

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

- Move the mouse over the breakpoint red circle, to be able to see the setting pop-up option (gear symbol) and click on it
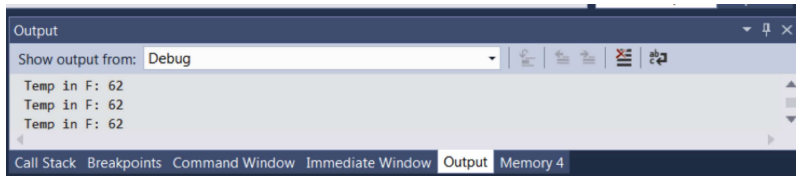
(/local--files/8avr:avradc/breakpoint2.png)

- Inside the **Breakpoint Setting Window**, check the **Actions** box. Inside the "Log a message to Output Window" insert the following: **Temperature in C = {Ctemp}** and check the **Continue execution box.** Click on **Close**
- Enter Debug mode by clicking on **Start Debugging and Break** in the top **Debug** menu
- Open the **Output Window** by selecting **Debug > Windows > Output** to see the value of the temperature variable
- Click on **Debug > Continue** and view the temperature in the Output window

✸ Results

The temperature of the chip is displayed in the output window. Pressing on it with a finger will heat up the reading by a couple of degrees.

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

7   **Task 7 - Disable debugWIRE and Close**

The **debugWire fuse** needs to be reset in order to program the Xplained board in the future. While still running in debug mode select **Debug > Disable debugWire and Close**. This will release the debugWire fuse.

🔍 Analysis

Using the ADC is quite easy and the debug feature makes it easy to monitor the results.

💡 Conclusions

This project can become the basis for future ADC related projects.