

Sensor de temperatura interna AVR

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

La medición de temperatura se basa en el sensor de temperatura en el chip que está acoplado a un canal ADC de un solo extremo. Seleccionar el canal ADC 8 escribiendo '1000' en ADMUX.MUX[3:0] habilita el sensor de temperatura. La referencia de voltaje interno de 1,1 V también debe seleccionarse para la fuente de referencia de voltaje ADC. Cuando el sensor de temperatura está habilitado, el convertidor ADC se puede usar en modo de conversión simple para medir el voltaje sobre el sensor de temperatura.

Muestra de datos de medición

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

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

Calibración

The results from temperature measurements have offset and gain errors. The internal temperature reference can be corrected for these errors by making calibration measurements at one or two known temperatures and adjusting the output values. This can result in very precise temperature measurements, sometimes as accurate as $\pm 2^\circ\text{C}$.

More detail can be found in this application note. (http://www.atmel.com/Images/Atmel-8108-Calibration-of-the-AVR's-Internal-Temperature-Reference_ApplicationNote_AVR122.pdf)

Configuring the ADC

The internal 1.1 V voltage reference must be selected for the ADC voltage reference source when using the internal temperature sensor. Writing "11" to the **REFS1** and **REFS0** bits of the **ADMUX** register selects the internal 1.1 V Voltage Reference.

The ADC has multiple input channels and modes of operation. The **Single Conversion** mode can be used to convert the temperature sensor signal connected to channel 8. To select channel 8, writing "1000" to the MUX3 thru MUX0 bits selects channel 8 or the temperature sensor.

Once the conversion is complete, the result is stored in two 8-bit ADC data registers ADCH (higher 8-bits) and ADCL (lower 8-bits). The 10-bit result can be either left justified or right justified. If ADLAR bit is set to a "1", then the result is left adjusted to the upper 10-bits of the two registers. If set to "0" then the result occupies the lower 10 bits of the two registers. By default, each bit is cleared and the word is right justified.

This code statement will set the bits as described.

```
ADMUX = (1<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (1<<MUX3) | (0<<MUX2) | (0<<MUX1) | (0<<MUX0);
```

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

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		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

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

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

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

Table 29-4 Input Channel Selection

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

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

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

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

Configuring the ADC Clock and Conversion Timing

The ADC can prescale the system clock to provide an ADC clock that is between 50 kHz and 200 kHz to get maximum resolution. If an ADC resolution of less than 10-bits is required, then the ADC clock frequency can be higher than 200 kHz. At 1 MHz it is possible to achieve up to eight bits of resolution.

The prescaler value is selected with **ADPS bits** in **ADCSRA Register**. For example; writing “110” to the **ADCSRA register** selects the divide by 64 pre-scaler resulting in a 125 KHz ADC clock when an 8 MHz oscillator clock is used.

ADC Control and Status Register A

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bit 7 – ADEN: ADC Enable

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

Bit 6 – ADSC: ADC Start Conversion

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

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

Bit 5 – ADATE: ADC Auto Trigger Enable

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

Bit 4 – ADIF: ADC Interrupt Flag

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

Bit 3 – ADIE: ADC Interrupt Enable

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

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

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

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

Table 29-5 Input Channel Selection

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

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

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

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

Starting a Conversion

In single conversion mode the **ADSC bit** in the **ADCSRA register** must be set to a logical one state to start the ADC conversion. This bit remains at logic high while the conversion is in progress and is cleared by the hardware, once the conversion is complete.

The first conversion after the ADC is switched on takes 25 ADC clock cycles in order to initialize the analog circuitry. Then, for further conversions, it takes 13 ADC clock cycles (13.5 for Auto-triggered conversions).

Sample Project

A sample project for using the Temperature Sensor is available here. (/8avr:avradc)

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 $\pm 10^{\circ}\text{C}$.

Material


Herramientas de hardware (opcional)

Herramienta	Sobre
 http://www.atmel.com/tools/MEGA328PB-XMINI.aspx	Mini kit de evaluación ATmega328PB-Xplained http://www.atmel.com/tools/MEGA328PB-XMINI.aspx

Herramientas de software

		Instaladores			
		Windows	Linux	Mac OSX	Instrucciones de instalación
Herramienta	Sobre				
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/EgVbV6l04yBAvXfl6BemPdoBdqWafKoruX9pkt21RgFG-w?e=pCb004

Additional Files

Files

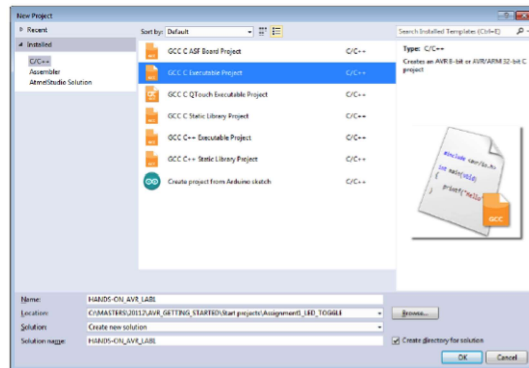


Xplained Board User Guide (http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf)

Procedure

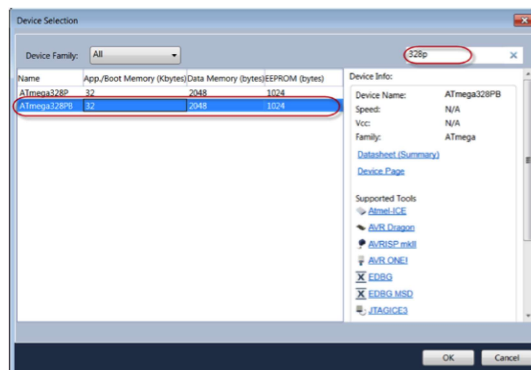
1 Task 1 - Project Creation

- 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.

```

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)
        ;

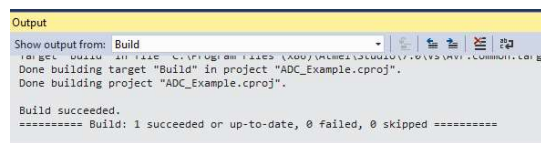
        /* Calculate the temperature in C */
        Ctemp = (ADC - 247)/1.22;
        Ftemp = (Ctemp * 1.8) + 32;
    }

    return -1;
}

```

3 Task 3 - Build Project

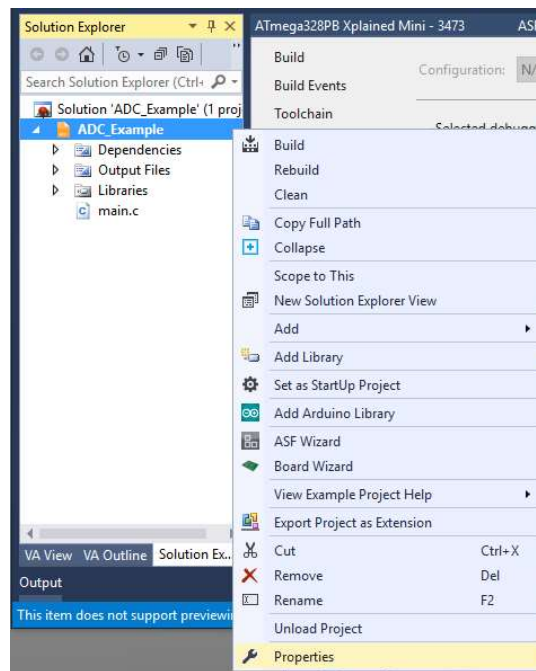
- Select **Build > Build Solution** from the Studio 7 menu to compile the code. You will see a **Build Succeeded** message in the output window. If there are any errors, check `main.c` for any mistakes in entering the program code.



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

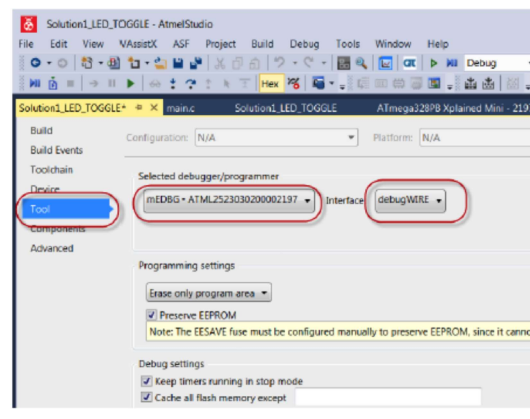
4 Task 4 - Programming the Xplained Board

- 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**.



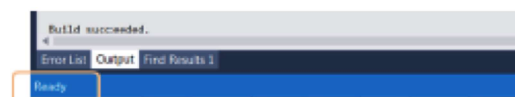
(/local--files/8avr:avradc/solution.png)

- 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.

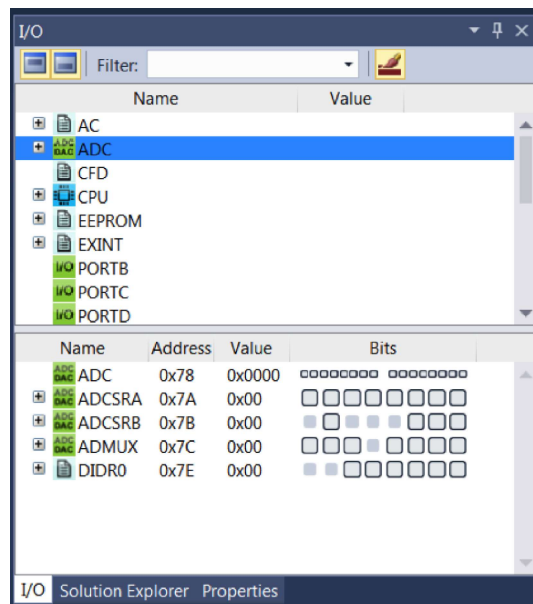


If the Xplained board will not connect, there may be a fuse setting causing this to occur (/boards:debugbrick).

5 Task 5 - Debugging

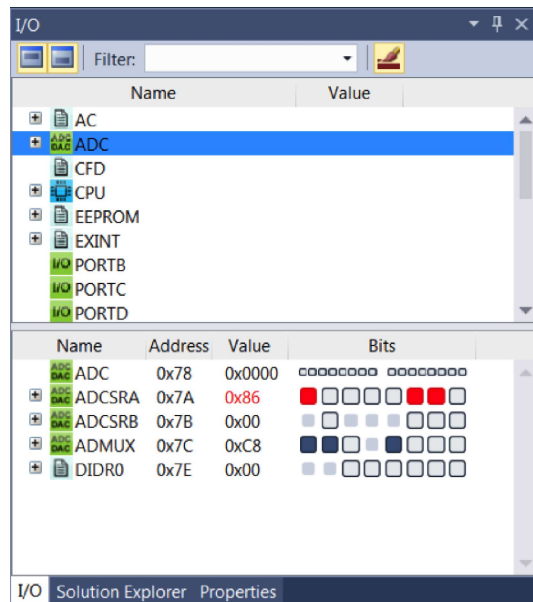
Debugging a device is essential for determining how a program may be running.

- 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)

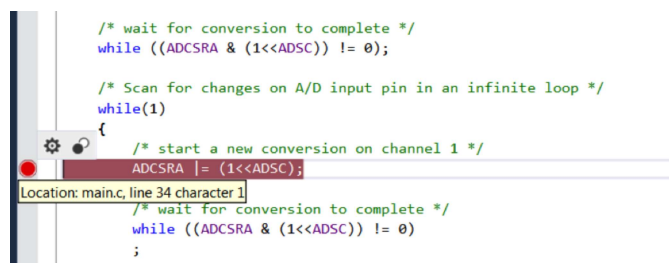
- Select **Debug > Step Over (F10)** to single-step through the program on the Xplained board. Monitor the ADC registers in the I/O View while single-stepping



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

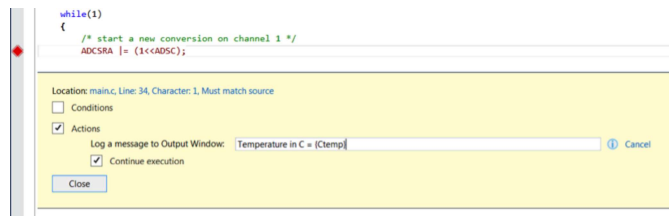
6 Task 6 - Breakpoint and Output

- 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

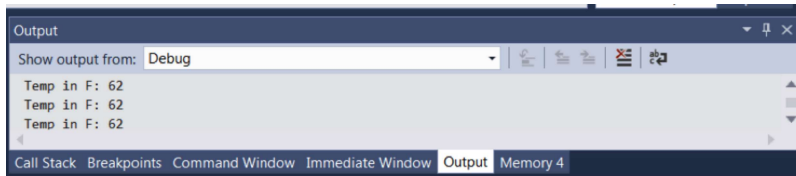


(/local--files/8avr:avradc/breakpoint2.png)

- 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.

Q 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.