Core Independent Peripherals on AVR

Core Independent Peripherals on AVR®

Prerequisites

- **Hardware Prerequisites**
  - Microchip ATmega328PB Xplained Mini board
  - Micro-USB cables (Type-A/Micro-B)
- **Software Prerequisites**
  - Atmel Studio 7.0
- **Required Knowledge**
  - Hands-on Training: AN17644: Getting Started with AVR Microcontroller
- **Estimated Completion Time: 30-60 minutes**

Introduction

This workshop will demonstrate how to build an application that reads the on-chip temperature sensor using an asynchronous interrupt handler, displays the temperature in Atmel Studio 7 debugger using the on-chip debug system, and optimize the application using core independent peripherals of the AVR. The core independent peripherals are designed to complete the tasks with no CPU intervention.

The following topics are covered:

- Combine Core Independent Peripherals in the ATmega328PB to automate tasks and allow the CPU to idle
- Implement asynchronous interrupt handlers for the Analog to Digital Converter (ADC) and use Timer/Counter event signal as ADC trigger source
- Use the ADC peripheral (found in most of the AVRs) to read the on-chip temperature sensor
- Toggle a LED using an I/O pin
- Use advanced debug features in Atmel Studio 7

Table of Contents

1. Icon Key Identifiers

The following icons are used in this document to identify different assignment sections and to reduce complexity.

Info: Delivers contextual information about a specific topic.

Tip: Highlights useful tips and techniques.

To do: Highlights objectives to be completed.

Result: Highlights the expected result of an assignment step.

Warning: Indicates important information.

Execute: Highlights actions to be executed out of the target when necessary.

2. Assignment 1: Peripheral Initialization

This assignment will create a real life application example, which combines AVR CIP - Core Independent Peripherals, AVR HW and SW tools, and Debugging features of the Atmel Studio 7.

In this project the ADC is used to read the on-chip temperature sensor every second, autonomously triggered by one of the timer/counters.

The ATmega328PB device features a 10-bit successive approximation ADC. The ADC is controlled by TIMER1 with it's capture event signal to trigger the ADC sampling.

As core is not involved to read the ADC, the use of such core independent peripheral technique is particularly helpful for time critical control-tasks.

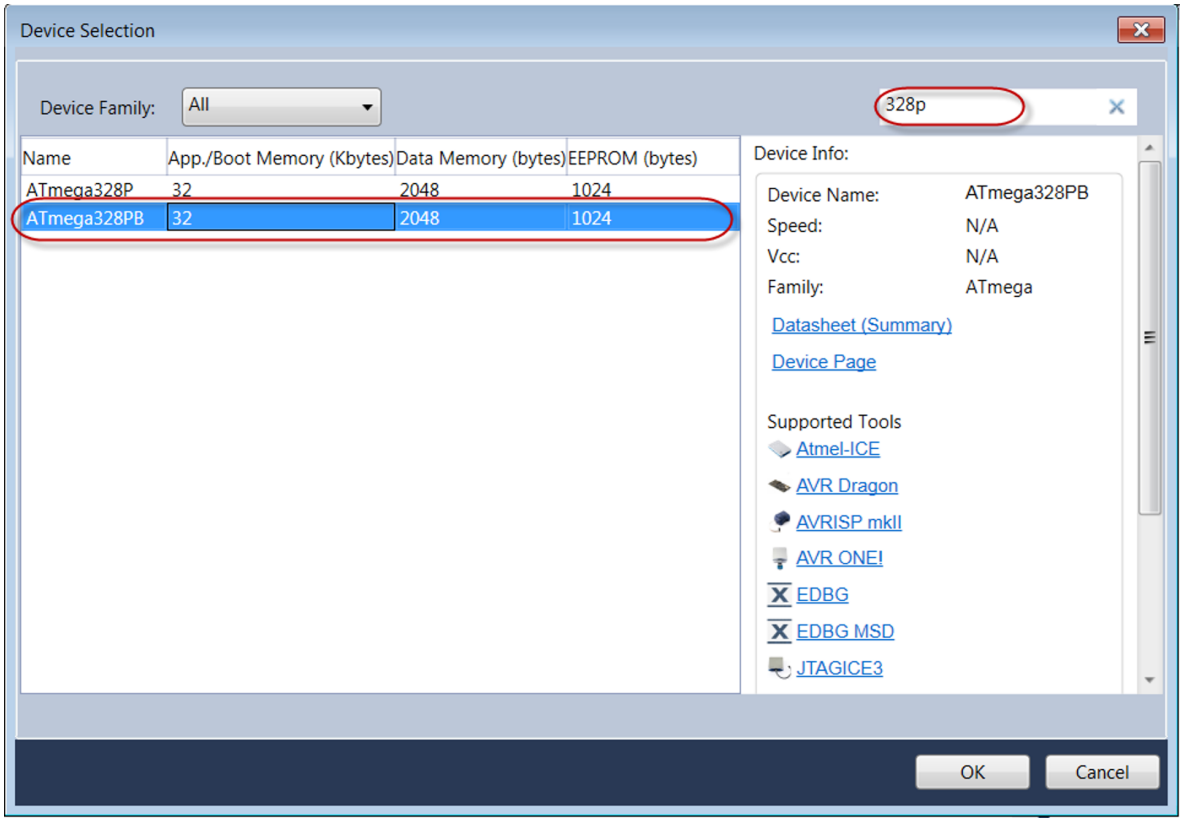
Application: Read temperature using channel 8 of the ADC every 1000ms. Focus on minimal usage of the CPU, which should be in sleep mode most of the time!

1. Project Creation

To do: Create a new Project.

1. Open **Atmel Studio 7**.
2. Select **File** → **New** → **Project**.
3. The New Project window appears. Select **GCC C Executable Project** and give the project 'Name' to “Assignment_CIP” and set the 'Location' (any path on PC) and Click **OK**.
4. Now the **Device Selection** window appears as shown in the following figure. In the search bar, type **328p**, then select the device Atmega328PB, and click **OK**.

Figure 2-1. Device Selection



Result: The AVR project is created.

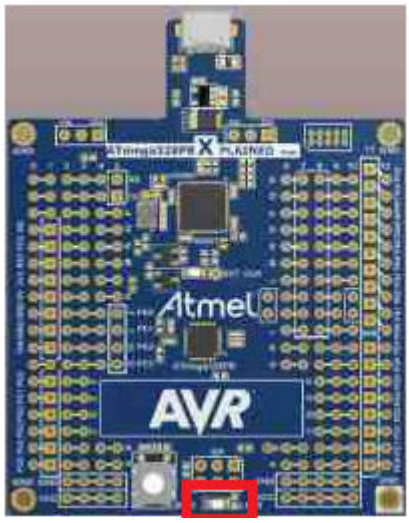
Add code below in main.c file and take a look at function `main()` in main.c file, which includes the LED setup and names of functions used to initialize the used peripherals.

```
#define ADC_ADMUX_REFS_INT_1_1V           3

#define ADC_ADMUX_TEMP                8

#define ADC_PRESCALER_DIV_128           7

#define ADC_ADCSRB_AUTO_TRIGGER_TC1_CE  7

#define TC_PRESCALER_DIV_1024           5


volatile int16_t temperature;

int main(void)
{
    // Set PORTB5 as output
    DDRB |= (1 << DDRB5);

       // Setup the ADC
    //InitADC();

    // Setup the Timer
    //InitTC1();

    //All code runs in asynchronous interrupt handler
    while (1)
```

```
    {
    ;
    }
}
```

Info: The on-board orange LED0 will be used to display the activity on the ATmega328PB Xplained Mini board. The LED0 is connected to pin PB5, a pin on PORTB, which can be checked in the ATmega328PB Xplained Mini board schematic.

Figure 2-2. LED0 on ATmega328PB Xplained Mini Board



2. Initialize the Analog to Digital Converter

To do: Insert the following code above the `main(void).`

```
static void InitADC(void)
{
       // Select internal 1.1 V reference, and tempsensor as input
    ADMUX = ( ??? << REFS0) | (??? << MUX0);

       // Set prescaler to 1024, enable Auto Trigger, enable interrupts(clear flag)
       // and enable ADC
    ADCSRA = (1 << ???) | (1 << ???) | (1 << ???) | (1 << ???) | (??? << ADPS0);

       // Set ADC trigger source to TC1 Capture Event
    ??? = (ADC_ADCSRB_AUTO_TRIGGER_TC1_CE << ADTS0);
}
```

Info: The ADC Block Diagram and description are given below.

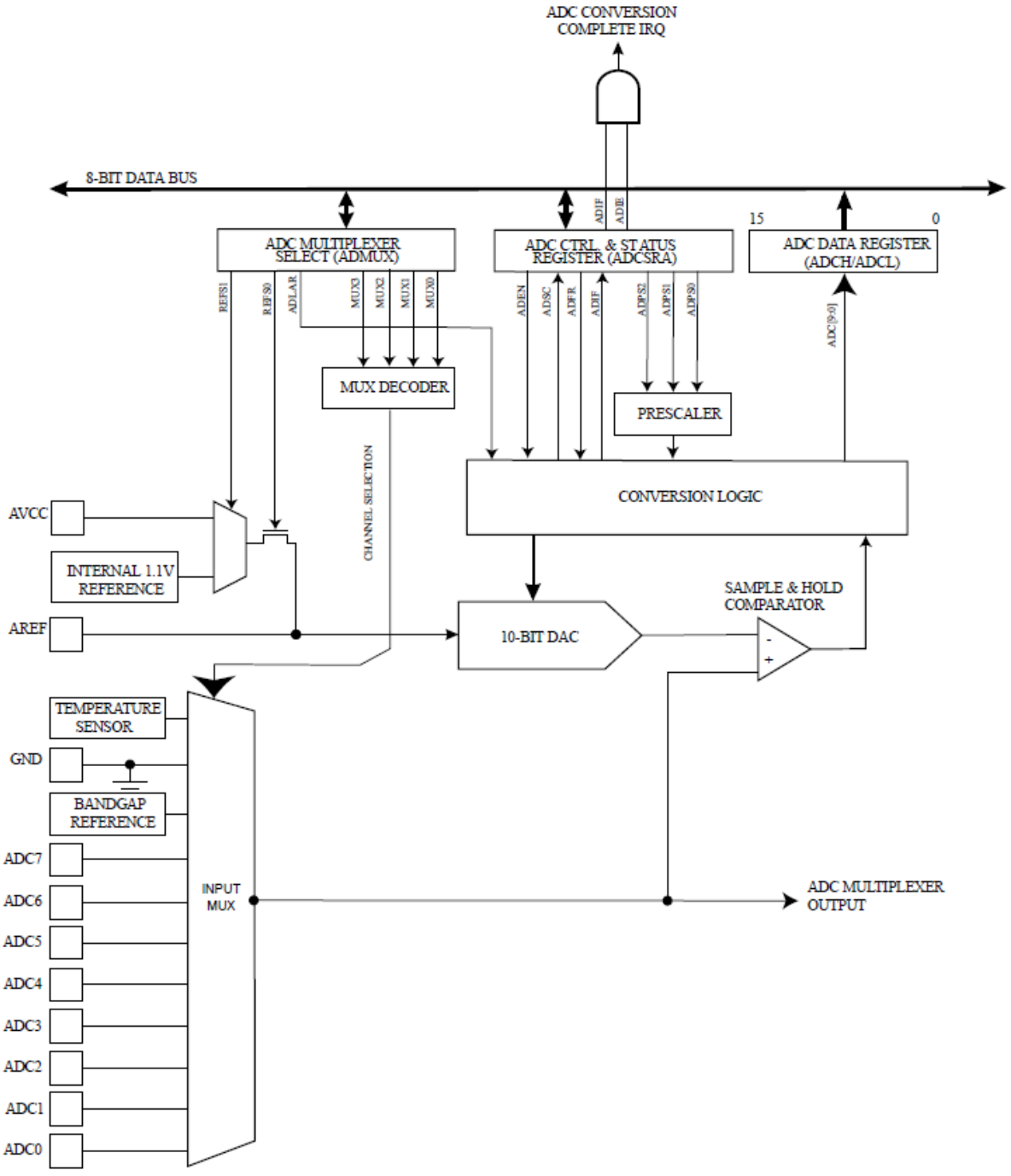
In the above function some of the code is missing. The missing code needs to be added as explained in the next steps:

a. Internal Reference.

b. Set up the ADCSRA.

c. Choose the trigger Source for the ADC.

Figure 2-3. ADC Block Diagram



The setup of the ADC is done by the following steps:

a. Internal Reference: Setup of the ADC involves configuring a stable reference voltage for measurements. The reference voltage for the ADC (VREF) indicates the conversion range for the ADC. Single ended channels that exceed VREF will result in codes close to 0x3FF. VREF can be selected as either AVCC, internal 1.1V reference, or external AREF pin. AVCC is connected to the ADC through a passive switch. The internal 1.1V reference is generated from the internal bandgap reference (VBG)

through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. VREF can also be measured at the AREF pin with a high impedance voltmeter. Note that VREF is a high impedance source, and only a capacitive load should be connected in a system.

The internal reference voltage of 1.1V offers sufficient precision needed for this application and is defined in the main.c file as below:

```
#define ADC_ADMUX_REFS_INT_1_1V 3
```

As the ADC has multiple input channels and modes of operation, in this case the Single Conversion mode will be used and configuration of the input channel to channel 8 is needed as defined in the top of the main.c file. Channel 8 will be read from the internal temperature sensor of the ATmega328PB. It is defined as below in main.c file.

```
#define ADC_ADMUX_TEMP 8
```

To do: Add the missing code to select the 1.1V Internal reference and to set the temperature sensor as input.

```
ADMUX = ( ??? << REFS0) | (??? << MUX0);
```

Tip: Use defined value ADC_ADMUX_REFS_INT_1_1V to set the internal reference value. Use defined value ADC_ADMUX_TEMP to set the temperature sensor as the ADC input.

Info: ADC Multiplexer Selection Register is shown in the figure below.

Figure 2-4. ADC Multiplexer Selection Register (this is a screen-shot from the ATmega328PB data sheet)

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

Bit 5 – ADLAR: ADC Left Adjust Result
The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see ADCL on page 325 and ADCH on page 326.

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
0110	ADC6
0111	ADC7
1000	Temperature sensor
1001	Reserved
1010	Reserved

b. Set up the ADCSRA - ADC Control and Status register A.

Info: This register is configured to enable the ADC (ADEN), Auto Trigger (ADATE), ADC Interrupt Flag (ADIF), Interrupts (ADIE), and to select Prescaler for the system clock (ADPSn).

Figure 2-5. ADC Control and Status Register A (this is a screen-shot from the ATmega328PB data sheet)

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

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
010	4
011	8
100	16
101	32
110	64
111	128

By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a resolution lower than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate. The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz.

The prescaling factor is selected by the 'ADC Prescaler Select bits' in the **ADC Control and Status Register A (ADCSRA.ADPS)**. The prescaler starts counting from the moment the ADC is switched ON by writing the ADC Enable bit ADCSRA.ADEN to '1'. The prescaler keeps running for as long as ADEN=1, and is continuously reset when ADEN=0.

To do: Add the missing code below.

```
ADCSRA = (1 << ???) | (1 << ???) | (1 << ???) | (1 << ???) | (??? << ADPS0);
```

Tip:

- Use defined value ADEN to enable the ADC
- Use defined value ADATE to make the ADC start a conversion on a positive edge of the selected trigger signal
- Use defined value ADIF to clear the Interrupt Flag
- Use defined value ADIE to activate the ADC Conversion Complete Interrupt
- Use defined value ADC_PRESCALER_DIV_128 to set the prescaler of the system clock to 128

c. Choose the trigger Source for the ADC.

The ADC don't need to run continuously as temperature measurements are done at a given configurable interval. The sample interval and the ADC is controlled by another autonomous peripheral, TIMER1, with it's capture event signal.

Info: For this purpose, bits 2, 1, and 0 are used in the register ADCSRB - ADC Control and Status register B.

Figure 2-6. ADC Control and Status Register B (this is a screen-shot from the ATmega328PB data sheet)

Bit	7	6	5	4	3	2	1	0
		ACME				ADTS2	ADTS1	ADTS0
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

Bit 6 – ACME: Analog Comparator Multiplexer Enable
When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see *Analog Comparator Multiplexed Input*..

Bits 2:0 – ADTSn: ADC Auto Trigger Source [n = 2:0]
If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 29-6 ADC Auto Trigger Source Selection

ADTS[2:0]	Trigger Source
000	Free Running mode
001	Analog Comparator
010	External Interrupt Request 0
011	Timer/Counter0 Compare Match A
100	Timer/Counter0 Overflow
101	Timer/Counter1 Compare Match B
110	Timer/Counter1 Overflow
111	Timer/Counter1 Capture Event

It can be seen from the screen-shot above that it is needed to configure value 0x111 to ADTSn bits.

To do: Add the missing code shown below to choose the correct register, where the Auto Trigger Source will be set.

```
??? = (ADC_ADCSRB_AUTO_TRIGGER_TC1_CE << ADTS0);
```

3. Initialize TIMER 1

To do: Insert the following code above the `main(void)` function.

```
static void InitTC1(void)
{
```

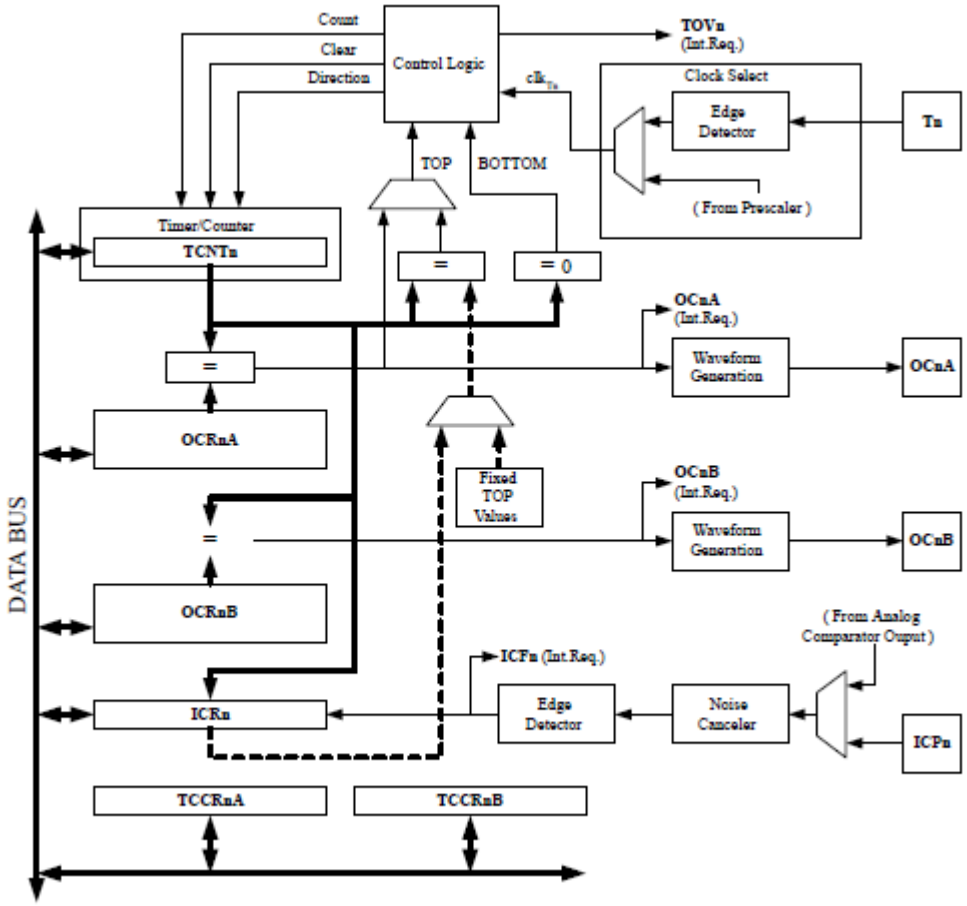
```
    // Set wave generation mode CTC with ICR1 as top value and prescaler to 1024
    TCCR1B = (1 << WGM12) | (1 << WGM13) | (TC_PRESCALER_DIV_1024 << CS10);

    // Set timeout to 1s
    ICR1 = ???;
}
```

Info: The 16-bit Timer/Counter Block Diagram and description is given below.

In the `InitTC1` function above some of the code is missing. The missing code needs to be added as explained below.

Configure Time interval to 1s

Figure 2-7. 16-bit Timer/Counter Block Diagram



The TIMER1 module is a 16-bit Timer/Counter unit that allows accurate program execution timing (event management), waveform generation, and signal timing measurement. In this exercise the timer is configured to reset the counter on reaching it's TOP value (CTC mode) and Input Compare Register (ICR1) is used to store the TOP value. Once the TOP value is reached, the counter will clear and restart. To achieve a 1 second time interval, the timer's built-in prescaler must be used.

Info: In this project the TOP value, or maximum Timer/Counter value, is defined by the **Input Capture Register 1 - ICR1 Register**.

The system clock is set to 16MHz and the prescaler is set to 1024. So, to achieve a 1 second time interval, we need to fill the ICR1 with value (16000000/1024) = 15625.

To do: Add the missing number shown below for the Input Capture Register 1 value.

```
// Set timeout to 1s
    ICR1 = ???;
```

Info: The Waveform Generation mode and Clock selection is configured in the TCCR1B register. The Waveform Generation mode is CTC mode, configured by writing bits WGM2 and WGM3 to 1. (Refer table **Waveform Generation Mode Bit Description** from the data sheet.) Clock prescaler CLKio/1024 is selected by configuring 'Bits 2:0 – CS[2:0]: Clock Select' in the TCCR1B register to 1. (Refer table **Clock Select Bit Description** from the data sheet.)

Result: The code included inside the application is shown below.

```
static void InitADC(void)
{
    // Select internal 1.1 V reference, and tempsensor as input
    ADMUX = (ADC_ADMUX_REFS_INT_1_1V << REFS0) | (ADC_ADMUX_TEMP << MUX0);

    // Set prescaler to 1024, enable Auto Trigger, enable interrupts (clear flag) and
       // enable ADC
    ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIF) | (1 << ADIE) |
(ADC_PRESCALER_DIV_128 << ADPS0);

    // Set ADC trigger source to TC1 Capture Event
    ADCSRB = (ADC_ADCSRB_AUTO_TRIGGER_TC1_CE << ADTS0);
}

static void InitTC1(void)
{
    // Set wave generation mode CTC with ICR1 as top value and prescaler to 1024
    TCCR1B = (1 << WGM12) | (1 << WGM13) | (TC_PRESCALER_DIV_1024 << CS10);

    // Set timeout to 1s
    ICR1 = 15625;
}

int main(void)
{
    // Set PORTB5 as output
    DDRB |= (1 << DDRB5);

       // Setup the ADC
    InitADC();

       // Setup the Timer
    InitTC1();

    while (1)
    {
    }
}
```

3. Assignment 2: Setup Core Independent Peripherals with Interrupt Handlers

At this moment the application is missing one important part; setting the ISR for the ADC.

The AVR provides several different interrupt sources. All interrupts including the Reset Vector have a separate program vector in the program memory space. All interrupts have assigned individual enable bit. In order to enable the interrupt you need to configure the enable bit at the peripheral to one together with the **Global Interrupt Enable bit in the Status Register**.

Info: Depending on the Program Counter value, interrupts may be automatically disabled, when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. In this application lock bits are not configured.

1. Include header file

To be able to use interrupts the user must add the header file where interrupts related macros and functions are defined.

To do: Include the header file at the top of main.c file.

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

Result: The file inclusion will look like as shown in the figure below.

Figure 3-1. Header File

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

2. Setup ISR for ADC

To do: Insert the following code before the `main(void)` function.

```
ISR(ADC_vect)
{
    // Calculate the temperature in C
    temperature = (ADC - ??? )/???;

    // Set PORTB5 low
    PINB |= (1 << PINB5);

    // Clear the input capture flag in the TC1
    TIFR1 = (1 << ICF1);
}
```

Info: An interrupt handler is by syntax an ANSI-C compliant function, embedded in a macro specific to the compiler. For GCC (used in Atmel Studio), the ISR() macro defines the interrupt handler. The ADC interrupt handler will only handle the interrupt trigger after a read and conversion is completed by the ADC. Remember that the configuration of the ADC defines the channel to read and the mode of operation for the ADC from previous assignment and there is no need to change this between reads and conversions.

To do: Read the information and tip below and add the missing code to the `ISR(ADC_vect)` function above.

Info: The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC8 channel. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor. The result can be accessed after the end of each conversion, as the interrupt is triggered and the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion the result is $ADC = \frac{V_{IN} \cdot 1024}{V_{Ref}}$ where V_{IN} is the voltage on the selected input pin and V_{Ref} is the selected voltage reference.

Tip: To calculate the temperature from the ADC output, use the formula below.

$T = \frac{ADC - T_{OS}}{k}$, where *ADC* is the ADC data register, *k* is a gain coefficient, and T_{OS} is the temperature sensor offset value.

Info: Calibration values for *k* and T_{OS} needs to be calculated as these values vary from one device to another.

Temperature sensor calibration can be referred from the AVR122 application note: http://ww1.microchip.com/downloads/en/AppNotes/Atmel-8108-Calibration-of-the-AVR.

The measured voltage is linearly proportional to the temperature as described in the following table. The voltage sensitivity is approximately 1mV/°C and the accuracy of the temperature measurement is ±10°C after calibration.

Figure 3-2. Temperature vs. Sensor Output Voltage (Typical Case)

Temperature	-40°C	25°C	+105°C
ADC	205LSB	270LSB	350LSB

From the table above, at 25°C, the expected ADC count is 270, so for 0°C it is 245.
Default calibration Values: T_{OS} = 245 and k = 1 (default gain).

In this example, select T_{OS} = 247 and k = 1.22. These chosen values are an example of the software calibration required in the conversion and will vary from one device to another.

So the temperature measurement formula will be:

```
// Calculate the temperature in C
temperature = (ADC - 247)/1.22;
```

PIN PB5 is connected to the LED0 on the ATmega328PB Xplained Mini board and by setting the PINB5 bit, this pin will toggle.

```
// Toggle PORTB5
PINB |= (1 << PINB5);
```

At the end of the ISR for the ADC, the Input Capture Flag (triggered by TIMER1) has to be cleared to ensure that TIMER1 will start counting again.

```
// Clear the input capture flag in the TC1
TIFR1 = (1 << ICF1);
```

3. Enable Global Interrupts

To do: Insert the following code into the `main(void)` function, before the `while()` loop to enable the global interrupt.

```
sei();
```

Result: The code should look like below.

```
ISR(ADC_vect)
{
    // Calculate the temperature in C
    temperature = (ADC - 247)/1.22;

    // Toggle PORTB5
    PINB |= (1 << PINB5);

    // Clear the input capture flag in the TC1
    TIFR1 = (1 << ICF1);
}

static void InitADC(void)
{
    // Select internal 1.1 V reference, and tempsensor as input
    ADMUX = (ADC_ADMUX_REFS_INT_1_1V << REFS0) | (ADC_ADMUX_TEMP << MUX0);

    // Set prescaler to 1024, enable interrupts (clear flag) and enable ADC
    ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIF) | (1 << ADIE) | (ADC_PRESCALER_DIV_128
<< ADPS0);

    // Set ADC trigger source to TC1 Capture Event
    ADCSRB = (ADC_ADCSRB_AUTO_TRIGGER_TC1_CE << ADTS0);
}

static void InitTC1(void)
{
    // Set wave generation mode CTC with ICR1 as top value and prescaler to 1024
    TCCR1B = (1 << WGM12) | (1 << WGM13) | (TC_PRESCALER_DIV_1024 << CS10);
```

```
    // Set timeout to 1s
    ICR1 = 15625;
}

int main(void)
{
    // Set PORTB5 as output
    DDRB |= (1 << DDRB5);

    // Setup the ADC
    InitADC();

    // Setup the Timer
    InitTC1();

    //All code runs in asynchronous interrupt handler
    sei();

    while (1)
    {
    }
}
```

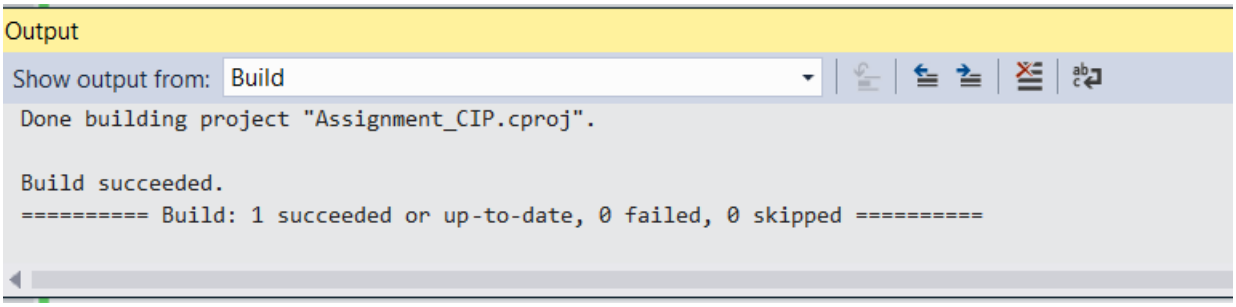
4. Assignment 3: Run and Debug the Application

Debug the application, Tips and Tricks!

To do: Compile the application using the F7 shortcut key. Alternatively, it can be done from the menu, **Build → Build Solutions**. The build should finish successfully with no errors.

Result: The result of the compilation is found in the output window in Atmel Studio 7 as shown below.

Figure 4-1. Build Solution



Info: The Flash and RAM footprint of the application are also found in the "Program Memory" and "Data Memory" usage in the 'Output' window.

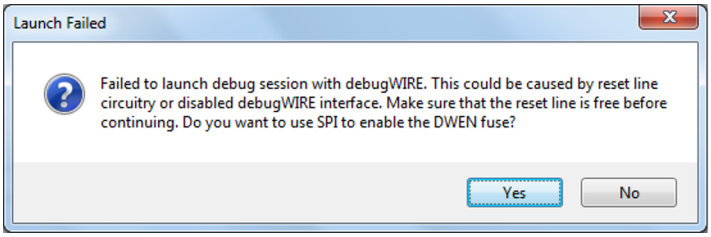
```
Task "RunOutputFileVerifyTask"
                Program Memory Usage     :    1004 bytes   3.1 % Full
                Data Memory Usage        :    2 bytes   0.1 % Full
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Assignment_CIP.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was
evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from
project "C:\MASTERS\20112\Assignment_CIP\Assignment_CIP\Assignment_CIP.cproj" (entry point):
Done building target "Build" in project "Assignment_CIP.cproj".
Done building project "Assignment_CIP.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

To do: Connect the ATmega328PB Xplained Mini board to the PC using a USB cable and start debugging by using the shortcut Alt+F5 or choose from menu **Debug → Start debugging and Break**

Info: If the 'Firmware upgrade' window is displayed, upgrade the firmware and close the 'Firmware upgrade' window.

Warning: If the DWEN fuse is not enabled an error message is displayed. Click **Yes** and the Atmel Studio 7 will use the SPI to set the fuse as shown below.

Figure 4-2. Enable the DWEN Fuse



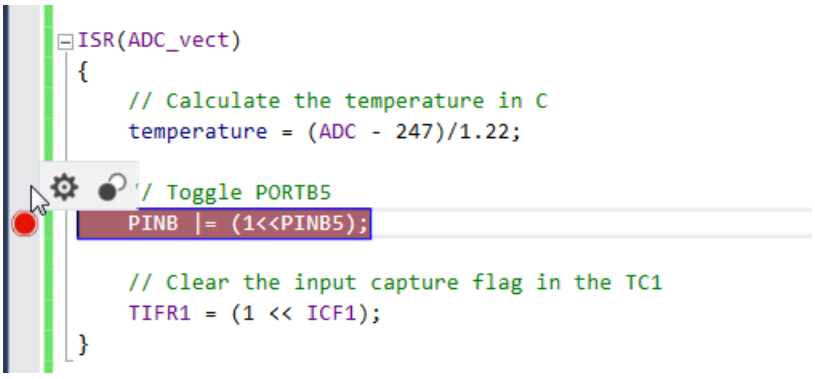
Result: This will start the debug session and stop on the first executable line of code in the `main()` function.

To do: Find the `ISR(ADC_vect)` function in the main.c file and set a breakpoint on the following line.

```
// Set PORTB5 low
PINB |= (1 << PINB5);
```

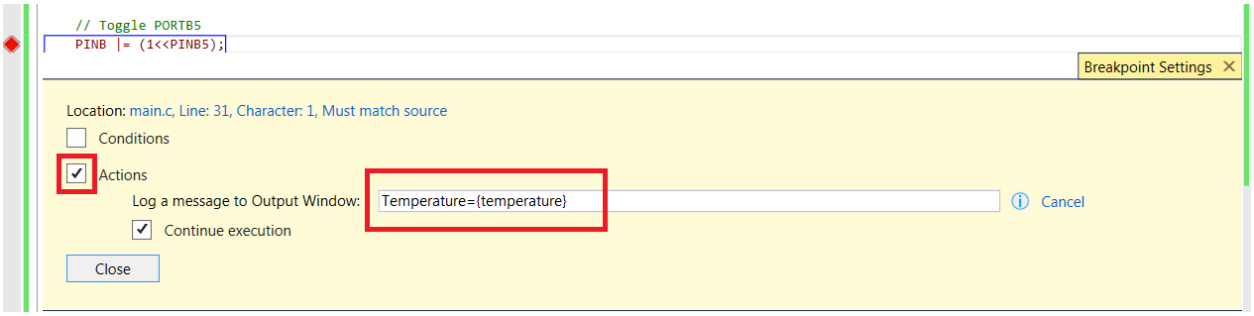
Info: Setting a breakpoint is done by using the F9 shortcut or from the menu, **Debug → Toggle Breakpoint**.

To do: Move the mouse over the breakpoint red circle to be able to see the **settings...** pop-up option. Click on the cogwheel icon.



To do: Inside the 'Breakpoint Setting' window, click to choose 'Actions' as shown below. Inside the 'Log a message to Output Window' insert the following: **Temperature = {temperature}** and hit 'Close'.

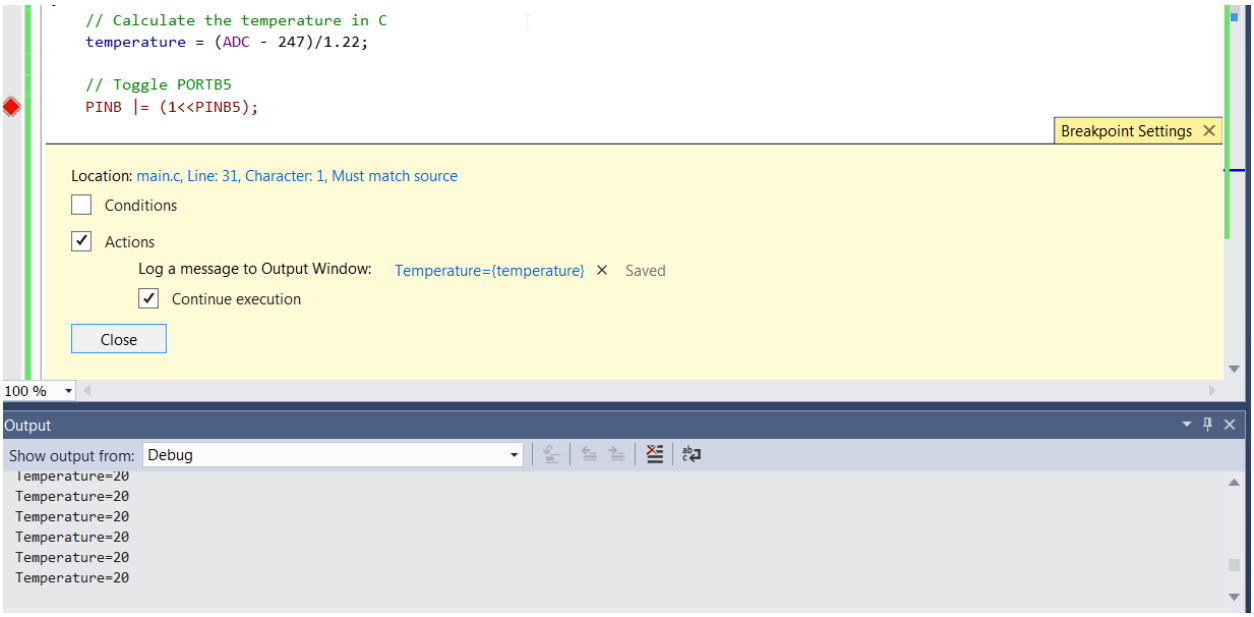
Figure 4-3. Setting the Output Window to Show Variable Temperature Value



To do: Start debugging by selecting **Debug → Continue**. Then open the 'Output' window by selecting menu **Debug → Windows → Output**.

Result: The value of the temperature variable is displayed in the 'Output' window as shown below.

Figure 4-4. Output Window: Temperature Display



Info: Click on **Debug → Break All** (Ctrl+F5) to stop the execution. Exit from debug mode by disabling debugWIRE. This is done by selecting menu **Debug → Disable debugWIRE and Close**.

5. Conclusion

This training demonstrated how to use core independent peripherals in an application to automate tasks and allow the CPU to idle.

Using the core independent peripheral technique is particularly helpful for time critical control-tasks as the AVR features a hardware interrupt controller that offloads work from the CPU.

With Atmel Studio 7 and advanced debugging feature it is easier to run real-time debugging of an application.

6. Revision History

Doc Rev.	Date	Comments
A	06/2017	Initial document release.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-1765-1

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	**Asia Pacific Office**	**China - Xiamen**	**Austria - Wels**
2355 West Chandler Blvd.	Suites 3707-14, 37th Floor	Tel: 86-592-2388138	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	Tower 6, The Gateway	Fax: 86-592-2388130	Fax: 43-7242-2244-393
Tel: 480-792-7200	Harbour City, Kowloon	**China - Zhuhai**	**Denmark - Copenhagen**
Fax: 480-792-7277	**Hong Kong**	Tel: 86-756-3210040	Tel: 45-4450-2828
Technical Support:	Tel: 852-2943-5100	Fax: 86-756-3210049	Fax: 45-4485-2829
http://www.microchip.com/	Fax: 852-2401-3431	**India - Bangalore**	**Finland - Espoo**
support	**Australia - Sydney**	Tel: 91-80-3090-4444	Tel: 358-9-4520-820
Web Address:	Tel: 61-2-9868-6733	Fax: 91-80-3090-4123	**France - Paris**
www.microchip.com	Fax: 61-2-9868-6755	**India - New Delhi**	Tel: 33-1-69-53-63-20
Atlanta	**China - Beijing**	Tel: 91-11-4160-8631	Fax: 33-1-69-30-90-79
Duluth, GA	Tel: 86-10-8569-7000	Fax: 91-11-4160-8632	**France - Saint Cloud**
Tel: 678-957-9614	Fax: 86-10-8528-2104	**India - Pune**	Tel: 33-1-30-60-70-00
Fax: 678-957-1455	**China - Chengdu**	Tel: 91-20-3019-1500	**Germany - Garching**
Austin, TX	Tel: 86-28-8665-5511	**Japan - Osaka**	Tel: 49-8931-9700
Tel: 512-257-3370	Fax: 86-28-8665-7889	Tel: 81-6-6152-7160	**Germany - Haan**
Boston	**China - Chongqing**	Fax: 81-6-6152-9310	Tel: 49-2129-3766400
Westborough, MA	Tel: 86-23-8980-9588	**Japan - Tokyo**	**Germany - Heilbronn**
Tel: 774-760-0087	Fax: 86-23-8980-9500	Tel: 81-3-6880- 3770	Tel: 49-7131-67-3636
Fax: 774-760-0088	**China - Dongguan**	Fax: 81-3-6880-3771	**Germany - Karlsruhe**
Chicago	Tel: 86-769-8702-9880	**Korea - Daegu**	Tel: 49-721-625370
Itasca, IL	**China - Guangzhou**	Tel: 82-53-744-4301	**Germany - Munich**
Tel: 630-285-0071	Tel: 86-20-8755-8029	Fax: 82-53-744-4302	Tel: 49-89-627-144-0
Fax: 630-285-0075	**China - Hangzhou**	**Korea - Seoul**	Fax: 49-89-627-144-44
Dallas	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	**Germany - Rosenheim**
Addison, TX	Fax: 86-571-8792-8116	Fax: 82-2-558-5932 or	Tel: 49-8031-354-560
Tel: 972-818-7423	**China - Hong Kong SAR**	82-2-558-5934	**Israel - Ra'anana**
Fax: 972-818-2924	Tel: 852-2943-5100	**Malaysia - Kuala Lumpur**	Tel: 972-9-744-7705
Detroit	Fax: 852-2401-3431	Tel: 60-3-6201-9857	**Italy - Milan**
Novi, MI	**China - Nanjing**	Fax: 60-3-6201-9859	Tel: 39-0331-742611
Tel: 248-848-4000	Tel: 86-25-8473-2460	**Malaysia - Penang**	Fax: 39-0331-466781
Houston, TX	Fax: 86-25-8473-2470	Tel: 60-4-227-8870	**Italy - Padova**
Tel: 281-894-5983	**China - Qingdao**	Fax: 60-4-227-4068	Tel: 39-049-7625286
Indianapolis	Tel: 86-532-8502-7355	**Philippines - Manila**	**Netherlands - Drunen**
Noblesville, IN	Fax: 86-532-8502-7205	Tel: 63-2-634-9065	Tel: 31-416-690399
Tel: 317-773-8323	**China - Shanghai**	Fax: 63-2-634-9069	Fax: 31-416-690340
Fax: 317-773-5453	Tel: 86-21-3326-8000	**Singapore**	**Norway - Trondheim**
Tel: 317-536-2380	Fax: 86-21-3326-8021	Tel: 65-6334-8870	Tel: 47-7289-7561
Los Angeles	**China - Shenyang**	Fax: 65-6334-8850	**Poland - Warsaw**
Mission Viejo, CA	Tel: 86-24-2334-2829	**Taiwan - Hsin Chu**	Tel: 48-22-3325737
Tel: 949-462-9523	Fax: 86-24-2334-2393	Tel: 886-3-5778-366	**Romania - Bucharest**
Fax: 949-462-9608	**China - Shenzhen**	Fax: 886-3-5770-955	Tel: 40-21-407-87-50
Tel: 951-273-7800	Tel: 86-755-8864-2200	**Taiwan - Kaohsiung**	**Spain - Madrid**
Raleigh, NC	Fax: 86-755-8203-1760	Tel: 886-7-213-7830	Tel: 34-91-708-08-90
Tel: 919-844-7510	**China - Wuhan**	**Taiwan - Taipei**	Fax: 34-91-708-08-91
New York, NY	Tel: 86-27-5980-5300	Tel: 886-2-2508-8600	**Sweden - Gothenberg**
Tel: 631-435-6000	Fax: 86-27-5980-5118	Fax: 886-2-2508-0102	Tel: 46-31-704-60-40
San Jose, CA	**China - Xian**	**Thailand - Bangkok**	**Sweden - Stockholm**
Tel: 408-735-9110	Tel: 86-29-8833-7252	Tel: 66-2-694-1351	Tel: 46-8-5090-4654
Tel: 408-436-4270	Fax: 86-29-8833-7256	Fax: 66-2-694-1350	**UK - Wokingham**
Canada - Toronto			Tel: 44-118-921-5800
Tel: 905-695-1980			Fax: 44-118-921-5820
Fax: 905-695-2078			