Before you start…

Here are some terms used in programming:

Program. A program is a set of instructions which the computer follows in processing data.

Text Editor. This is an application used to type or encode your program

Source Code. A source code is your program written in a programming language like C

Object Code. This is your program in a language that machines can read and understand

Compiler. A compiler translates your source code into an object code.

Compiler Directives. This are instructions to the compiler made at the start of the code.

Function. A function is a set of instructions that are often used in a program.

Library. Several functions commonly used in C are included in libraries.

Exercise 1: C Programming Basics

In this exercise, you will create your first C program. You will learn the basic structure of a C code as well as how to save, compile and run a program.

1. Type the code below in the text editor.

```
#include <stdio.h>

int main(void){
}
```

2. Save the code using the filename main.c.

3. This program does not do anything but it should compile. To compile, type `# gcc main.c -o main` in the command line. Be sure that the filename and object file is correct (`main.c` is the filename and `main` is the object file).

4. Now, add the line `printf("Hello World!\n");` to the main function. The new code should look like this:

```
/****** Hello World Program ******/

#include <stdio.h>

int main(void){
      printf("Hello World!\n");
      return 0;
}
```

5. Save and compile the code.

6. Run the code by typing `# ./main` in the command line. What is displayed on the screen?

7. The `printf` function is used to display text on the screen. Modify the program to display your name.

Notes: Take note that each line in the code, except for the compiler directives, is terminated with a semicolon. Also, make sure that all parentheses, braces, brackets and quotation marks are paired. Lines starting with "`//`" or enclosed by "`/*`" and "`*/`" are considered as comments and are disregarded by the compiler.

Exercise 2: Variables and Constants

In this exercise, you will learn how to use variables and constants. We will use a program that computes the area and circumference of a circle as an example.

1. Compute the area and circumference of a circle with a radius of 2. We will use these values to check the program.

2. Type the code below in the text editor.

```
/*************************** Pi Program ****************************/
/* Computes the area and circumference of a circle given the radius. */

#include <stdio.h>
#define PI 3.14159

int main (void){
       int radius = 2;
       float area;
       float circ;

       // Calculate the area and circumference
       area = PI*radius*radius;
       circ = 2.0 * PI * radius;

       // Display the results
       printf("Radius = %d\n",radius);
       printf("Area = %f\n", area);
       printf("Circumference = %f\n", circ);

       return 0;
}
```

3. Save and compile the code. You may use any filename.

4. Run the program.

5. What is the area and circumference displayed by the program? Do these match your computed values?

6. Try changing the value of the variable radius.

7. Try changing the value of the constant PI.

Notes: It is good practice to write a pseudocode of your program first. This way you have an outline to follow while creating your code. This is particularly helpful in making long and complicated codes.

Exercise 3: Getting User Input

Accepting user input is important in creating user interfaces. In this exercise, you will use the `scanf` function to get different types of input.

1. Type the code below in the text editor.

```
#include <stdio.h>
#define CURRENT_YEAR 2008

int main(void){
      int age;
      int year;

      printf("Enter your age: ");
      scanf("%d", &age);

      year = CURRENT_YEAR - age;
      printf("You were born in %d!",year);

      return 0;
}
```

2. Save and compile the code.

3. Run the program.

4. Modify the Pi Program to compute the area and circumference of a circle with a radius given by the user.

Notes: Variables used as parameters of the `scanf` function are preceded with the address operator, "`&`".

Exercise 4: Control Structures

1. Run a program containing the piece of code below.

```
int i = 10;

while(i > 0){
      printf("%d\n", i);
      i--;
}
```

   What is the output?

2. Run a program containing the piece of code below.

```
int i = 10;

do{
      printf("%d\n", i);
      i++;
}while (i > 0);
```

   What is the output?

3. Change the initial value of i to 0 in the code from #1. What is the output?

4. Change the initial value of i to 0 in the code from #2. What is the output?

5. Run a program containing the piece of code below.

```
int i;

for(i = 10; i > 0; i--){
      printf("%d\n", i);
}
```

   What is the output?

6. Modify the Pi Program to display the area and circumference of circles with radii from 1 to 10.

Exercise 5: Logical Test Operations

1. Run a program containing the code snippet below.

```
if (a < 10){
      printf("a is less than 10");
}
else if (a < 100){
      printf("a is greater than 10 but less than 100");
}
else {
      printf("a is greater than 100");
}
```

2. Modify your code in #1. Use a switch statement instead of if-then-else statements.

Exercise 6: Arrays and Strings

1. Run a program containing the code snippet below.

```
int i, n[5];

for (i = 0; i < 5; i++){
      printf("Enter number %d", i);
      scanf("%d", &n[i]);
}
```

2. Use the code in #1 to create a program that accepts 5 integers then displays the numbers in reverse order.

Exercise 7: Functions

1. In your main() function, declare an array of 5 integers.
2. Create a function getnum() that accepts 5 integers from the user.
3. Call the getnum() function in your main() function.
4. Create a printnum() function that prints the 5 integers.

Exercise 8: Resistance Calculator

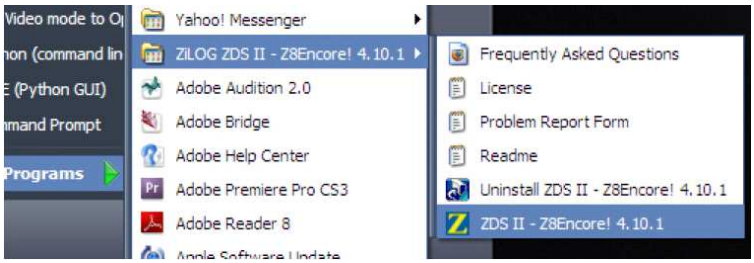
Create a program that computes the value of parallel resistances. It should:

1. Ask the user how many parallel resistances are there
2. Ask the user to input the value of the resistances as integer one by one
3. Display the resulting resistance value

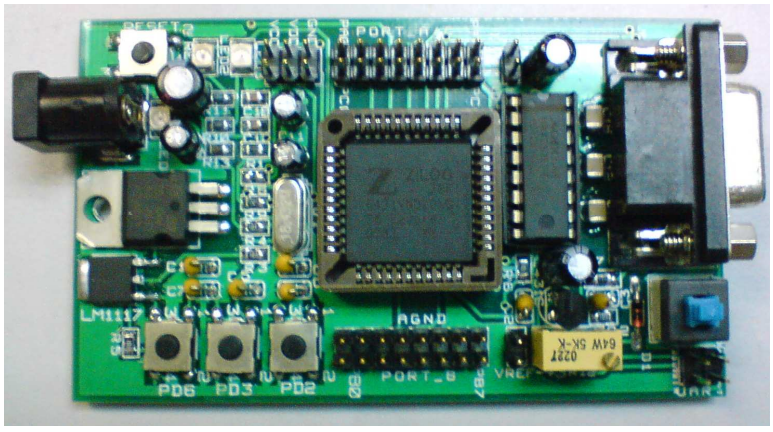
Exercise 9: Slimboard Getting Started

Before you can use the Slimboard, you have to check if you have the following:

1. ***PC/Laptop installed with Zilog Development Studio II.*** Look for the icon on the desktop or look for ZDS II – Z8 Encore! on the Start Menu.



2. ***Slimboard with microcontroller.*** Below is a picture of the Slimboard. At the center of the board is the Z8 Encore! microcontroller chip.



3. ***AC/DC Adapter.*** The adapter is used to power up the Slimboard. It converts the 220V ac from the outlet to 9V dc. Be sure to check the rating of the adapter and the outlet.

4. ***Serial Cable or USB-to-Serial Cable.*** If you are using a pc, you have a serial port and you will need a serial cable to connect the Slimboard to the pc. If you don't have a serial port, you will need a USB-to-Serial cable to connect the Slimboard's serial port to a USB port.


Serial Cable

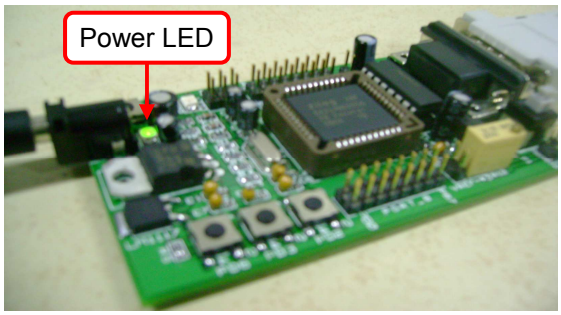

USB-to-Serial Cable

Exercise 10: Slimboard Beginner's Guide

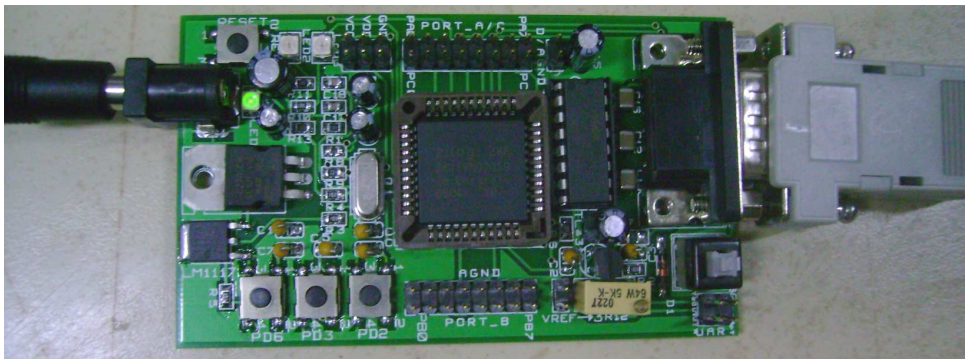
In this exercise, you will be familiarized with using the Slimboard. You will be guided on how to set-up and program the microcontroller.

Preparing the Slimboard

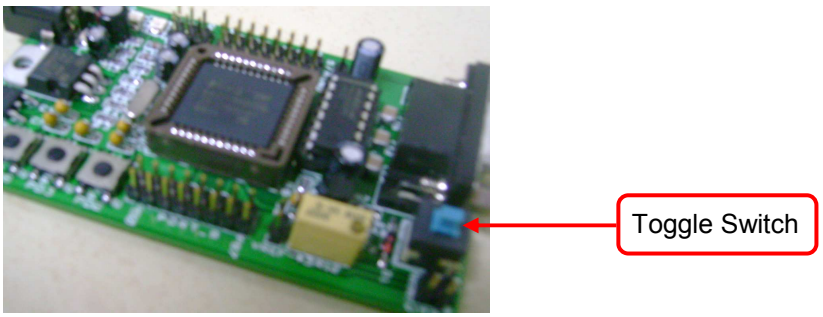
1. Power up the Slimboard using the ac/dc adapter. The Power LED should light up.



2. Connect the Slimboard to the serial port of the pc using the serial cable.

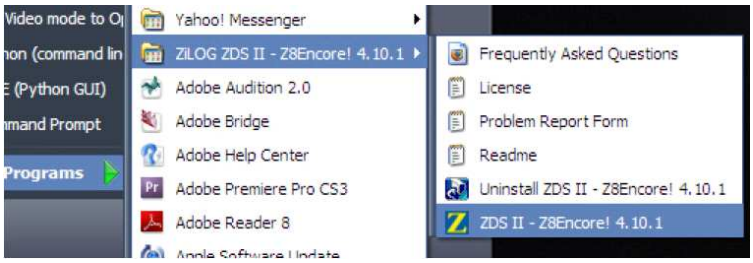


3. The serial port on the Slimboard is used in two ways – (1) DEBUG mode for programming the microcontroller and (2) UART mode for communication. Check the toggle switch near the serial port on the Slimboard. This should be high for DEBUG mode and low for UART mode.



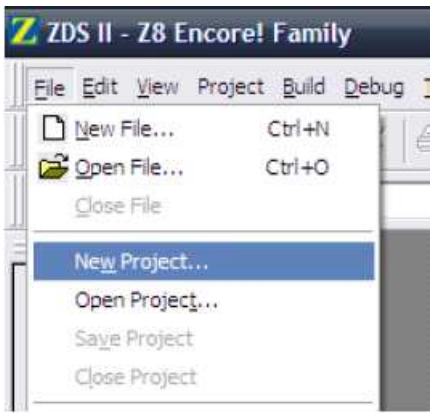
Programming the Microcontroller

1. Open ZDS II

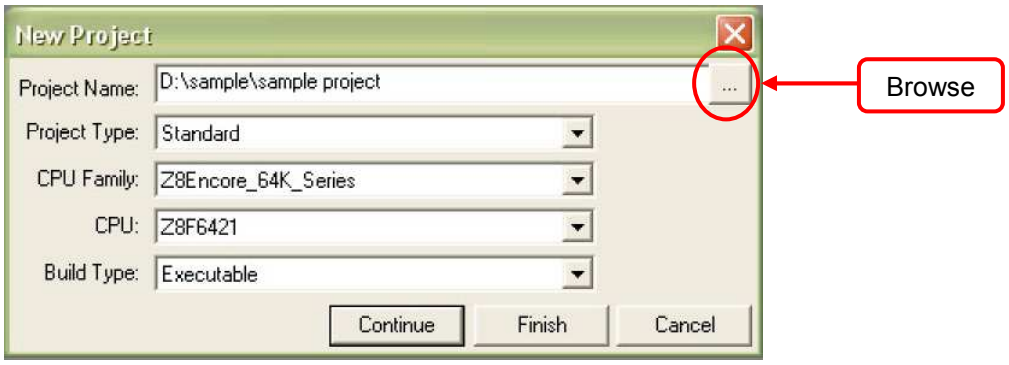


2. A project contains all the files required to create the firmware for your microcontroller. Create a new project by selecting

   `File > New Project`



3. You should see a window similar to the one shown below. Browse for the project location then enter the project name. It is advisable to create a new folder to contain the files in your project. In the figure below, “sample” is the project folder and sample project is the “project name”.



Fill-in the following data:

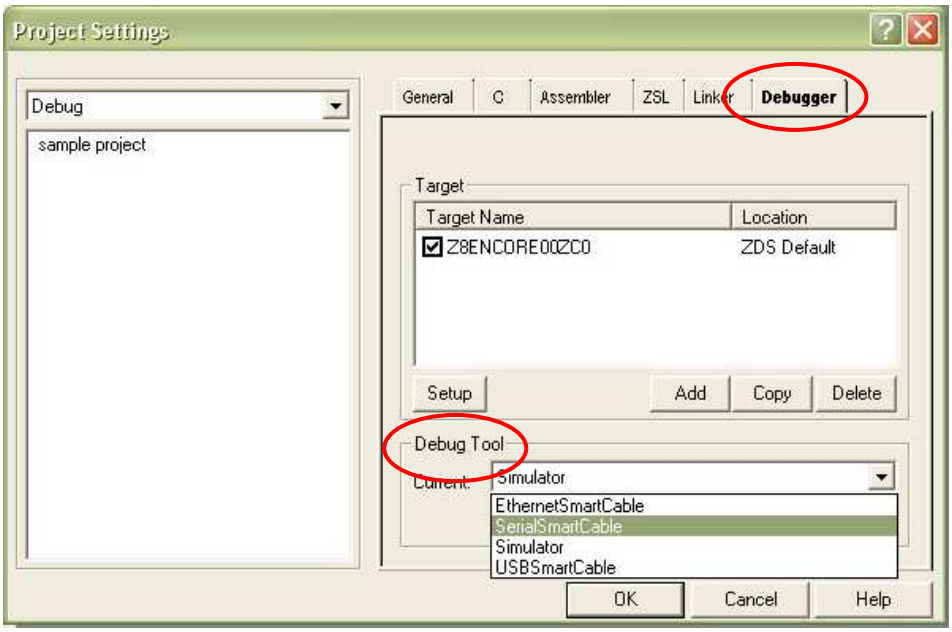
Project type: Standard
CPU Family: Z8Encore_64K_series
CPU: Z8F6421
Build Type: Executable

Click the Finish Button.

4. From the main menu select

   `Project > Settings`

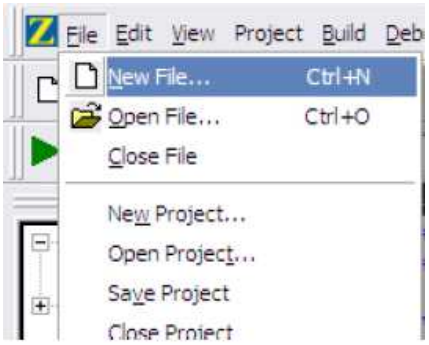
   Click the `Debugger` tab. On the `Debug Tool` box, select `Serial Smart Cable` from the dropdown menu. Then click the `Setup` button.

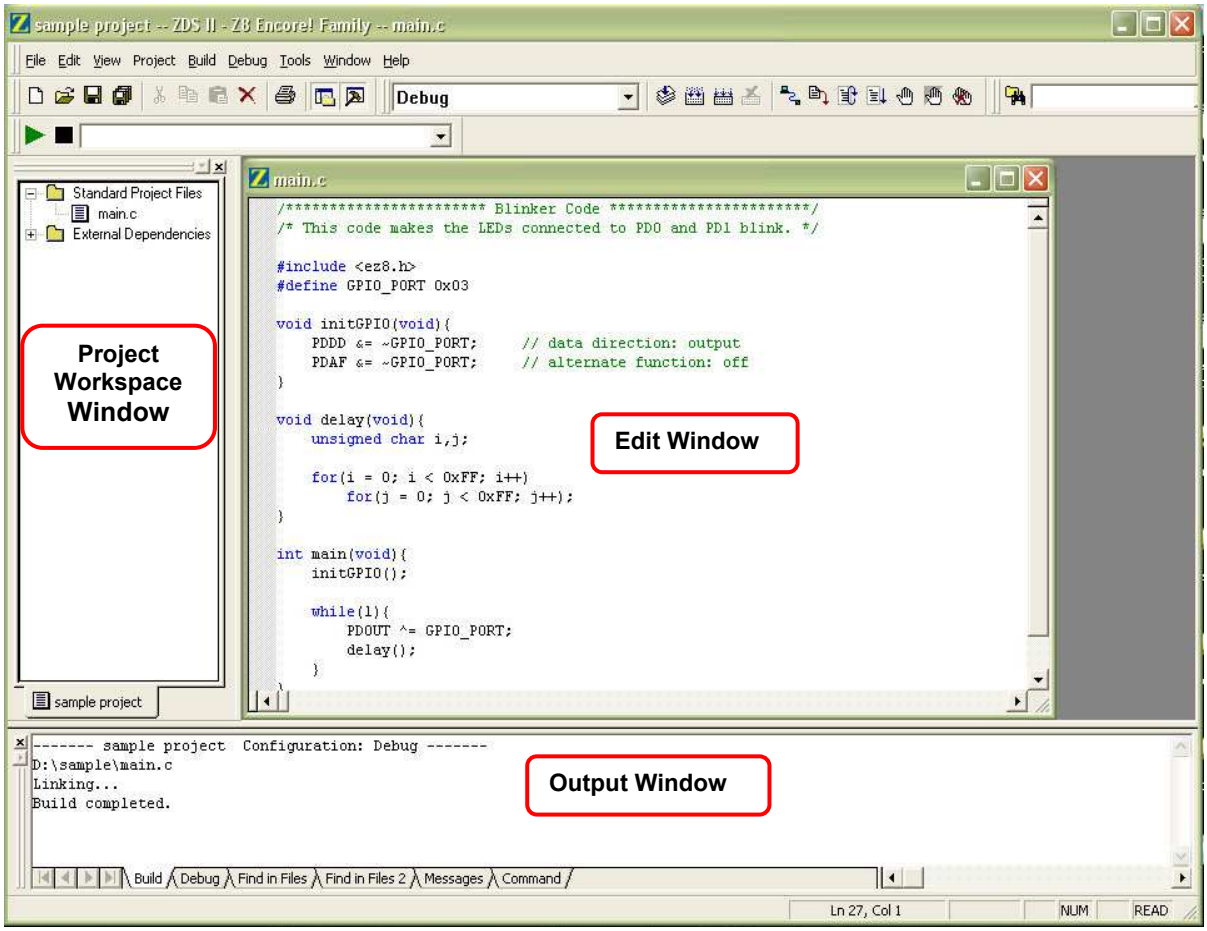


5. A dialog box similar to the one below should appear. For the `Baud Rate`, choose 57600 then select the `Port` connected to your Slimboard. Click the `OK` button on the `Setup Serial Communication` box. Click the `OK` button on the `Project Settings` window.

6. You are now ready to write your code. From the main menu, select

`File > New File`





ZDS II Developer's Environment

7. A text editor should appear. This is the Edit Window where you type your C code. For now, just copy the Blinker Code given below.

```
/********************** Blinker Code **********************/
/* This code makes the LEDs connected to PD0 and PD1 blink. */

#include <ez8.h>
#define GPIO_PORT 0x03

void initGPIO(void){
       PDDD &= ~GPIO_PORT;        // data direction: output
       PDAF &= ~GPIO_PORT;        // alternate function: off
}

void delay(void){
       unsigned char i,j;

       for(i = 0; i < 0xFF; i++)
              for(j = 0; j < 0xFF; j++);
}

int main(void){
       initGPIO();

       while(1){
              PDOUT ^= GPIO_PORT;
              delay();
       }
}
```

8. At this point your code is not yet part of the project. Click on the `Standard Project Files` folder in the project workspace window. Select `Add Files To Project`.

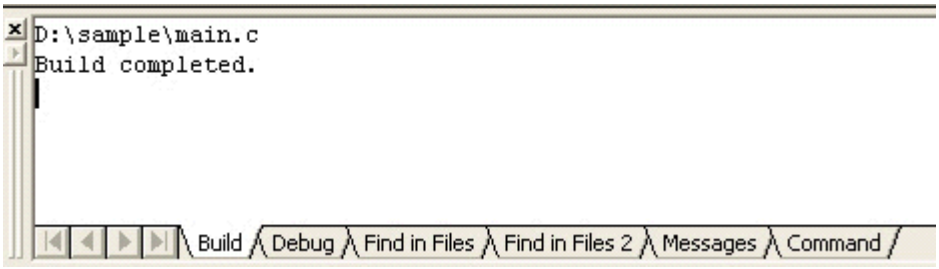
An `Add Files To Project` window should appear. Browse for and select the C files that you want to include in your project.

Notes: This exercise will serve as your guide in programming the Z8 Encore! microcontroller. Everything is pretty much the same. All you have to change is the C code. Your project may contain several C files. Save the file containing the main function as main.c.

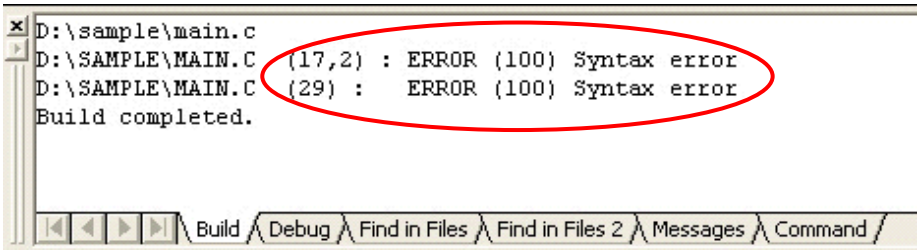
The added files should appear under the `Standard Project Files` folder.

9. Compile individual codes by clicking on the `Compile/Assemble File` button.

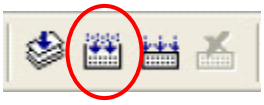
The results should be displayed on the output window at the bottom of ZDS.



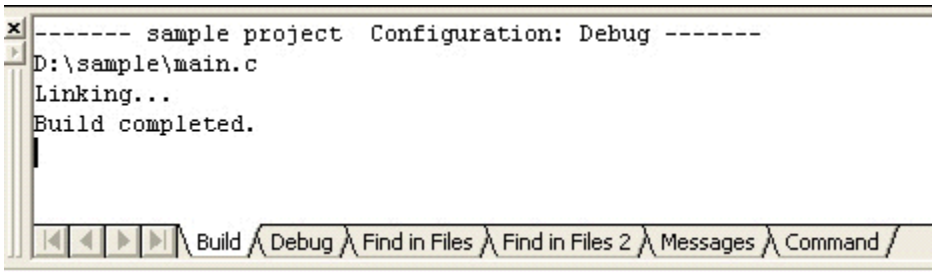
If the code does not compile successfully, the errors should also be shown in this window. Double-click on the error to move the text editor cursor to the line with error.



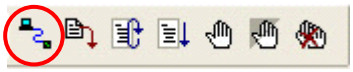
10. Build the project by clicking the `Build` button.



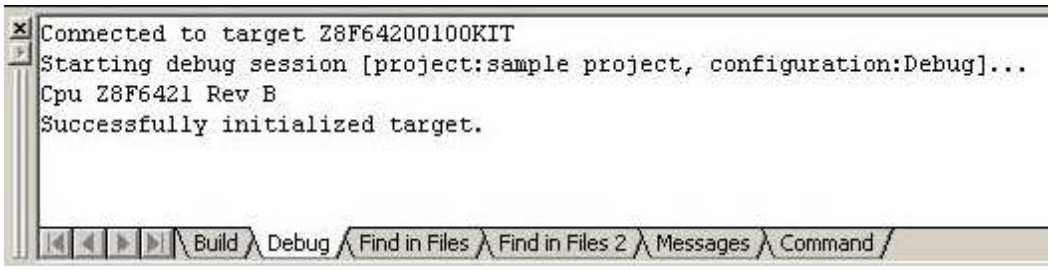
Again, the results should be displayed on the output window.



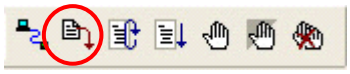
11. You may check if your pc can connect properly to the Slimboard by clicking the Connect To Target button.



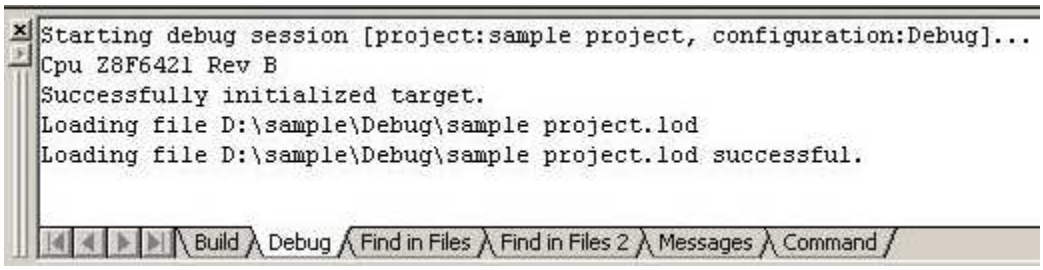
The result will be displayed on the output window.



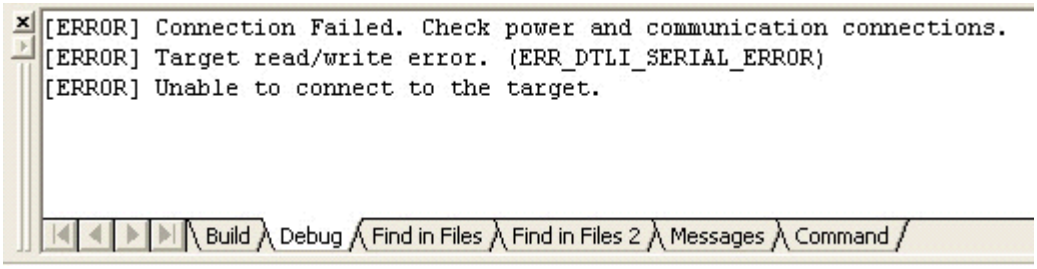
12. After building your program, you can now download it to the microcontroller by clicking on the Download Code button.



The output window should display the status after attempting to download.



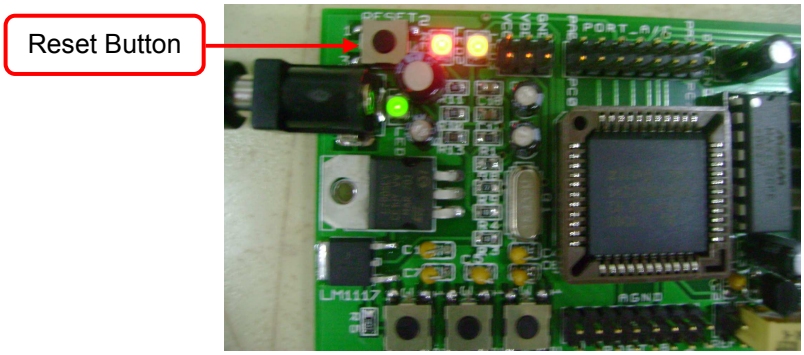
Some errors may occur during downloading.



If there are errors you may check the following:

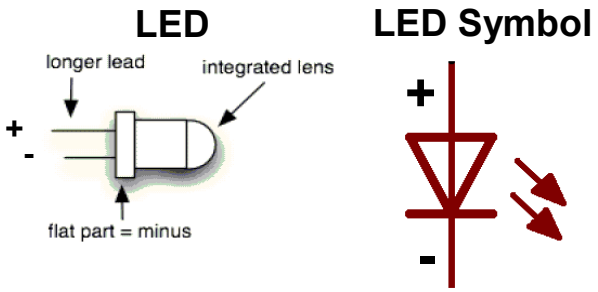
a. The Slimboard is connected properly to the serial port of the pc.
b. The correct COM port or serial port is selected.
c. The toggle switch on the Slimboard is in high position, indicating DEBUG mode.
d. No other application is using the serial port connected to the Slimboard.

13. Press the Reset Button on the Slimboard to check if your program works!



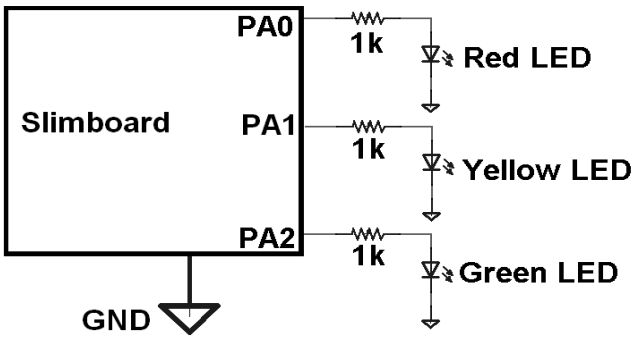
Exercise 11: GPIO

Electricity flows in a diode if the voltage on the positive side is greater than the voltage on the negative side. This is the ON state of the diode. The other case is of course the OFF state in which the diode is "open". An LED or Light Emitting Diode becomes a light source when it is in the ON state. Shown below are LEDs and the LED symbol. Always take note of the polarity of diodes when you use them in your circuit.



You have already downloaded the Blinker Code that makes the on-board LEDs of the Slimboard to blink. This time, you will learn the basic LED circuit and how to connect it to your Slimboard. You will also learn the different ways to use the GPIO pins of the Slimboard.

1. Setup the circuit below on your protoboard.



2. Modify the Blinker Code to

    a. enable high-drive mode on PA0, PA1 and PA2, and
    b. make the red, yellow and green LEDs blink simultaneously

3. Download the modified code to your microcontroller. Exercise #10 may serve as your guide in programming the microcontroller.

4. Using the pushbutton connected to PD2 of the Slimboard, make a program that toggles the state (on/off) of the LEDs with each press. Pressing once should turn on the LEDs, pressing again should turn off the LEDs, and so on.

5. Create a stoplight system that works in the following manner

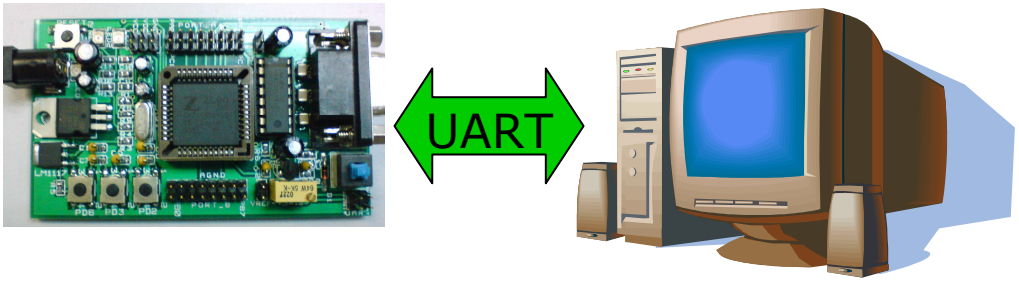
    a. default state: RED

    b. on pushbutton press

        i. green light for 10 delays
        ii. orange light for 2 delays
        iii. back to red default

Exercise 12: UART

UART stands for **Universal Asynchronous Receiver/Transmitter**. It is one way electronic devices, such as your computer and microcontroller, communicate with each other. This exercise demonstrates how to create a program that enables you to send characters to your computer. It also shows how to use the HyperTerminal which is an application that can read and display data from the serial port.



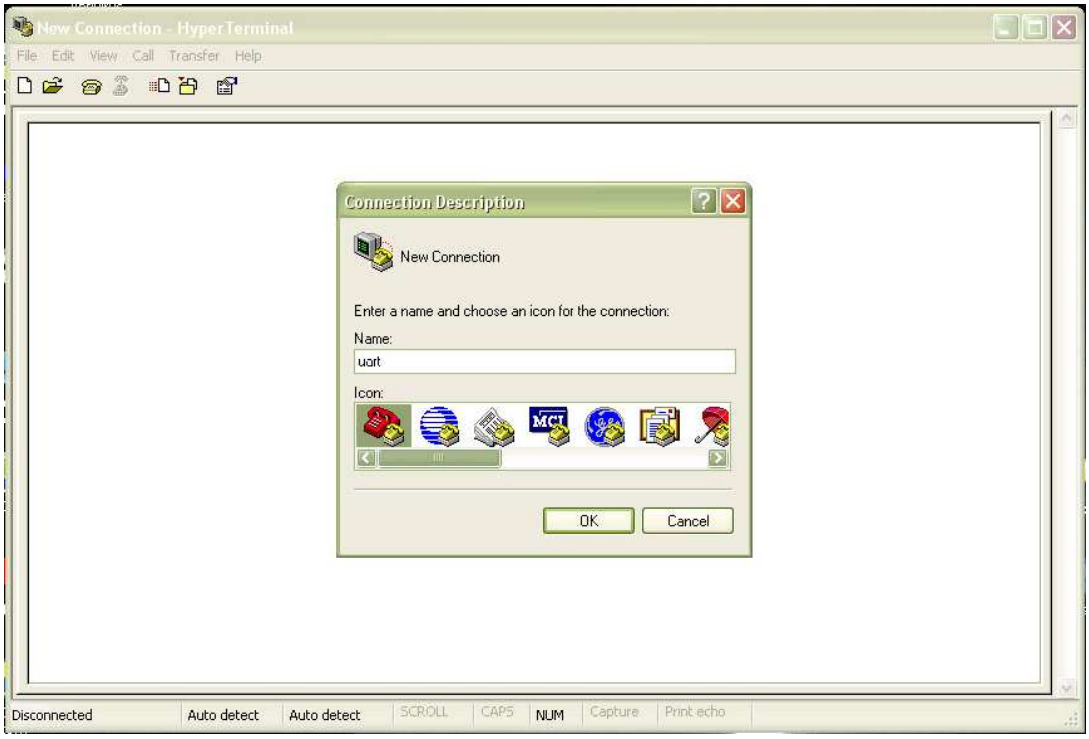
1. Download the code below to your microcontroller.

```
/************************ Hello World! ************************/
/*  This program displays a line of text on the HyperTerminal. */

#include <ez8.h>
#include <stdio.h>
#include <sio.h>

int main(void){
      init_uart(_UART1, _DEFFREQ, _DEFBAUD); // initialize UART0

      while(1){
            printf("Hello World!\n");        // display
      }
}
```

2. Open the HyperTerminal from the Start Menu of your computer. Select

   `All Programs > Accessories > Communications > HyperTerminal`

3. A `Connection Description` window should appear. Enter any name for the session. Click the `OK` button.

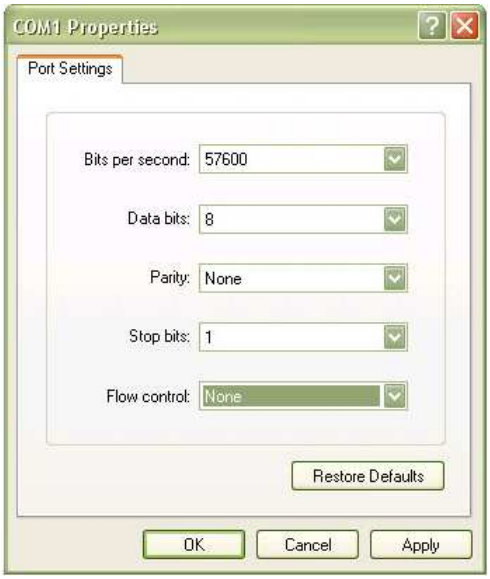


4. The `Connect To` window should appear. Choose the correct serial port on the `Connect using` dropdown menu. Then click the `OK` button.

5. A window containing the properties of the selected port should appear. Fill in the following data:

Bits per second:	57600
Data bits:	8
Parity:	None
Stop bits:	1
Flow control:	None

   Click the OK button.



6. Check if the toggle switch on the Slimboard is in High Position indicating UART Mode. Press the Reset button on the board.

7. What is displayed on the HyperTerminal?

Exercise 14: Timers – Blinker

You have already created a Blinker Program in the GPIO Exercises. However, the specified delay did not directly translate to time as we know it – in seconds! This time, you will create a blinker program in which you can set the delay using timers.

1. Download the code below to your microcontroller.

```
/************************ Blinker Using Timers ***********************/

#include <ez8.h>
#include <sio.h>

int counter=0;

#pragma interrupt          // routine function
void isr_timer0(void){
       if (counter==4){
              counter=0;
              PDOUT ^= 0x01;
       }
       else counter++;
}

void init_timer0 (void){
       DI();

       T0CTL = 0x79;              // initialize timer 0111 1001, continuous
                                  // mode with 128 prescaler
       T0H = 0x00; T0L = 0x01;
       T0RH = 0x8C; T0RL = 0xA0;  // the reload value

       SET_VECTOR(TIMER0,isr_timer0);
       IRQ0ENH |= 0x20;
       IRQ0ENL |= 0x20;
       IRQ0 &= ~0x20;


       EI();
}

void main (void){
       PDDD &= ~0x01;
       PDAF &= ~0x01;
       PDOC &= ~0x01;
       PDHDE |= 0x01;


       init_timer0();
       T0CTL |= 0x80;             // turn on timer
       while(1);
}
```

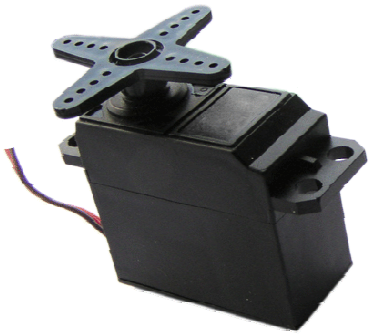
3. What happens?

4. Change the lines in red into

```
T0RH = 0x46
T0RL = 0x50
```

5. What happens?

Exercise 15: Timers – Servo Motors



1. Setup the servo motor by connecting
    a. the RED wire to the positive pole of the battery,
    b. the ORANGE wire to PA1 of the Slimboard,
    c. the BROWN wire to GND, and
    d. the negative pole of the battery to GND.

2. Download the code below to your microcontroller.

```
/********************* Servo Motor Control *********************/

#include <ez8.h>
#include <sio.h>

void init_timer0 (void){
       PAAF |= 0x02;                      // turn on alternate function

       T0CTL = 0x7B;                      // initialize timer 0111 1011, PWM
                                          // mode with 128 prescaler
       T0H = 0x00; T0L = 0x01;
       T0RH = 0x06; T0RL = 0xC0;          // reload value of the period
       T0PWMH = 0x00; T0PWML = 0x48;      // reload value of the on-time period
}

void main (void){
       init_timer0();


       T0CTL |= 0x80;                     // turn on timer
       while(1);
}
```

3. What happens?

4. Change the lines in red into

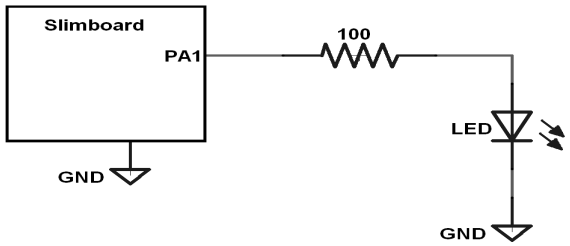
```
T0PWMH = 0x02
T0PWML = 0xD0
```

5. What happens?

Exercise 16: Timers

1. Modify the Blinker Using Timers Code to make the LEDs on the Slimboard blink every 1 second.

2. Setup the circuit below on your protoboard.



3. Modify the code in #1 to change the intensity of the LED when the push button connected to PD2 is pressed.

4. Setup your servo motor again.

5. Make a program that will make the servo motor

   a. rotate 45 degrees clockwise if you press the push button connected PD2 and

   b. rotate 45 degrees counter clockwise if you press the push button connected to PD3

Exercise 17: Light Sensors

There are several kinds of light sensors. In this exercise, you will be using Light Dependent Resistors (LDR). An LDR is a sensor that changes its resistance with respect to the amount of light that strikes its surface. As the intensity of light increases, the resistance decreases

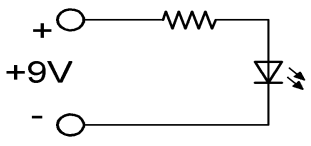
As you go along in this exercise, fill-in the table below.

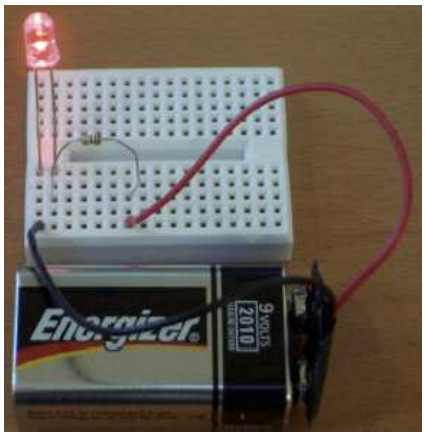
Light	LDR resistance
Ambient	
No Light	
Superbright Light	
Red	
Green	
Blue	

1. Expose the LDR to ambient light. Measure the LDR resistance.

2. Cover the LDR such that it can not sense light. Measure the LDR resistance.

3. Set up the circuit below on your protoboard. Use a Superbright White LED.





4. Expose the LDR to the superbright light. This set-up should be enclosed. Measure the LDR resistance.

5. Repeat #3 and #4 using red, green and blue LEDs.

Exercise 18: Temperature Sensors

For this experiment you will be using the following:

Thermistor

LM35

Thermistor. The thermistor is a sensor that varies its resistance with temperature. As temperature increases, thermistor resistance decreases.

LM35. An LM35 is an integrated circuit temperature sensor. Its output voltage is directly related to change in temperature.

As you go along in this exercise, fill-in the table below.

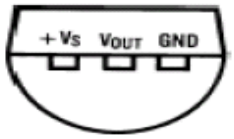
	Temperature Using Thermometer	**Thermistor Resistance**	**LM35 Output Voltage**
Room			
Body			

1. Using a thermometer, measure the room temperature and your body temperature.

2. Measure the resistance of the thermistor at room temperature.

3. Place the thermistor inside your clasped hand. Measure the resistance of the thermistor.

4. Power up the LM35 from a 5V supply. Make sure that the +Vs pin is connected to the positive supply and the GND pin is connected to ground. The pin outs are shown here.



BOTTOM VIEW

5. Do #2 and # 3 using the LM35.

6. Using the values from the table, make a plot of Temperature vs Thermistor Resistance and Temperature vs LM35 Output Voltage.

7. From the plots, derive the transfer function of the thermistor and the LM35.

8. Try to put the thermistor under/right below your nostril (where inhaled and exhaled air passes). What happens to the thermistor resistance as you inhale? as you exhale?

9. Do #8 using the LM35.

	Thermistor Resistance	**LM35 Output Voltage**
Inhalation		
Exhalation		

Exercise 19: Colorimeter

Making a Color Sensor

Light has 3 primary colors: Red, Green, and Blue. A mix of 2 or more in varying intensities can produce other colors. For instance, Red and Green combined produce yellow. Reflected light can be sensed by the LDR. If we know the color of the light being reflected from an object, then we can deduce the color of the object itself. Let's say I have a red object. When I flash Green or Blue light, both colors get absorbed by the object, but when I flash red light, the "Red" color is reflected almost completely! Hence, when we want to know the color of the object, we can employ the use of the LDR and measure the resistance when each of the Red, Green, and Blue lights is flashed individually. The more the LDR resistance changes when a specific light is flashed, the more positive we are that the color of the object must contain a component of the light reflected(i.e. when Green is flashed and the LDR resistance does not change, then we can conclude that the object absorbs green light).

1. Measure the LDR's resistance when Red light (Red LED) is flashed directly onto the LDR(about an inch apart). Repeat using Green and yellow LEDs.

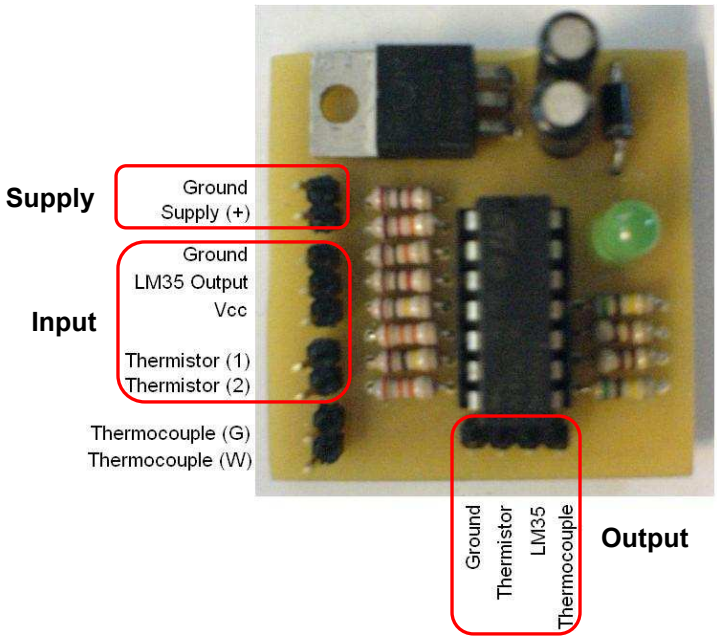
2. Try the SuperBright LEDs and read the LDR resistance for each LED flashed directly onto the LDR.

3. Now try to "bounce" the light off the object whose color we wish to "read" /decipher. Read the LDR resistance for each reflected light of each object (try Red, Yellow, and Green) when the superbright LED is flashed one at a time.

Light	LDR Resistance
Red	
Green	
Yellow	
Superbright Red	
Superbright Green	
Superbright Yellow	
Reflected Red	
Reflected Green	
Reflected Yellow	

Exercise 20: TempEST

The thermistor changes its resistance but the LM35 changes its voltage when temperature varies. Most, if not all, microcontroller systems "prefer" reading voltages rather than resistances. The **TempEST** or **Temperature-to-Electric Signal Sensing Tool** board acts as signal conditioning board to translate thermistor resistance change to voltage change.



1. Connect the thermistor and LM35 to the input side of the TempEST.

2. Power up the TempEST using a 9V battery.

3. Using a thermometer, measure the room temperature and your body temperature.

4. Measure the output voltage for the thermistor at room temperature.

5. Place the thermistor inside your fist or clasped hand. Measure the output voltage for the thermistor.

6. Do #4 and # 5 using the LM35.

	Temperature Using Thermometer	**Thermistor Output**	**LM35 Output**
Room			
Body			

Exercise 21: Analog-to-Digital Conversion

1. Download the code below to your microcontroller.

```
#include <ez8.h>
#include <stdio.h>
#include <sio.h>

#define CHANNEL 0x00             // ANA0
#define MODE 0x00                // Singleshot (Continuous = 0x10)
#define VREF 0x00                // Internal VRef (External = 0x20)
#define ENABLE 0x80              // CEN

void InitADC(void){
       PBAF = 0x01;              // PB0 = ANA0
       ACTL = (CHANNEL | MODE | VREF);
}

int PollADC(void){
       ACTL |= ENABLE;           // Enable ADC
       while (ACTL & ENABLE);    // Wait for End of Conversion
       return (((int)ADHR << 2) | (ADLR >> 6));
}

void delay(void){
       unsigned char i,j;

       for(i = 0; i < 0xFF; i++)
              for(j = 0; j < 0xFF; j++);
}

void main(void){
       int VALUE = 0;
       float volts = 0.0;

       init_uart(_UART0, _DEFFREQ, _DEFBAUD);

       printf("Analog-to-Digital Conversion\n");
       InitADC();
       while(1) {
              VALUE = PollADC();
              volts = VALUE / 512.0;     // Turn ADC Readng to Volts
              printf("ADC: %X, %4.3f volts \n",VALUE, volts);
              delay()
       }
}
```

2. Open the HyperTerminal. Follow the steps in Exercise 12.

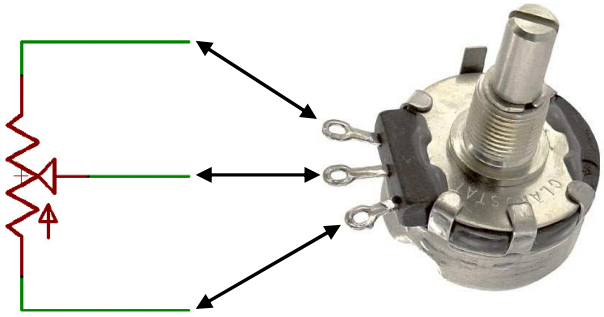
3. Press the reset button.

4. What is the ADC value displayed on the HyperTerminal?

5. Touch the PB0 pin. What is the ADC value this time?

Exercise 22: Voltmeter

In this exercise, you will use a potentiometer. A potentiometer is a resistor with variable resistance. You can change the resistance between two adjacent leads of a potentiometer by turning the knob. The resistance between the leads at the ends of the potentiometer is constant. This is the value or rating of the potentiometer.

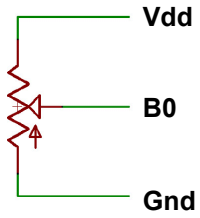


Potentiometer Symbol

Potentiometer

1. Do Exercise 21.

2. Connect the potentiometer to the Slimboard as shown below.



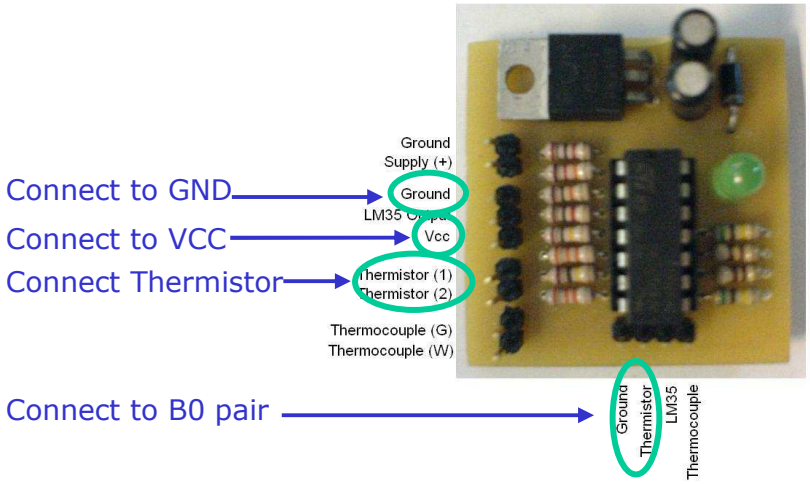
3. Press Reset on the Slimboard.

4. What happens on the displayed ADC values as you turn the knob of the potentiometer?

Exercise 23: ADC (TempEST)

You are already familiar with the TempEST board. As you have learned, it translates the output of the temperature sensors into voltage levels that can be accepted by the microcontroller as inputs. In this exercise, you will use ADC to read the output voltages of the TempEST board.

1. Connect the TempEST board to the Slimboard as shown below.



2. Use the ADC code to read the voltage output for the thermistor.

3. Record the displayed voltage at room temperature and at body temperature.