Introduction to A/D conversion and the MSP432

Introduction - Analog Vs. Digital

An analog signal is continuous in amplitude and time; that is, it can take on any value at any point in time, as shown on the curve in the left portion of Figure 2.4.1. A digital signal, on the other hand, is discrete in both amplitude and time, and it can have only a finite range of values it can take on, as shown on the right in Figure 2.4.1.

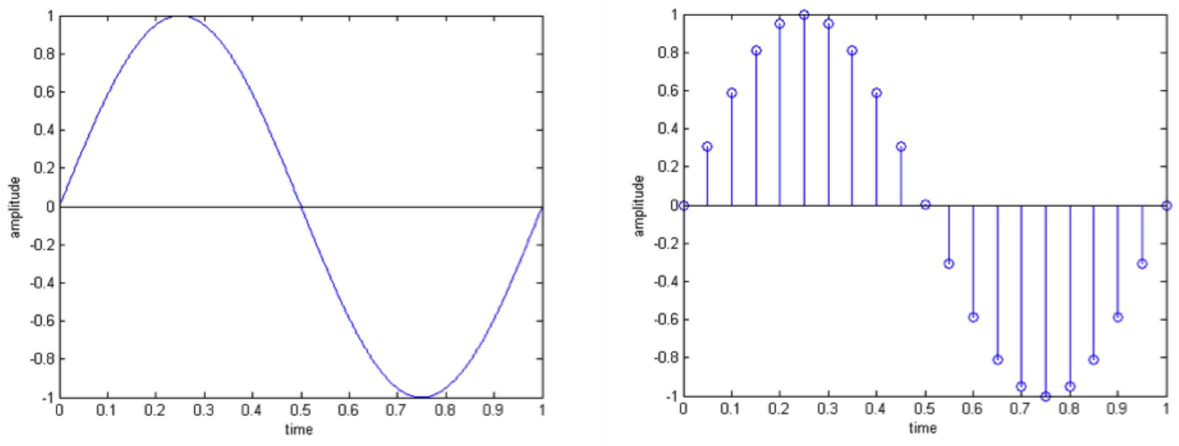


Figure 2.4.1: An analog signal on left is continuous in both time and amplitude, and a digital signal on right has discrete values for both time and amplitude.

Analog to Digital Conversion

A computer is only able to work with discrete (digital) signals. However, almost all physical, real-world values are actually continuous (analog) signals. For example, when a microphone records a sound, the sound is a continuous or analog signal. As the sound propagates through air it manifests in changes to air pressure, these changes are converted to the movements of a diaphragm within the microphone. Various methods are then used to turn the vibrations into electrical signals (variant currents or voltages). However, before a computer can read this signal, the signal must be formatted so that it is a quantized, discrete valued function (readable by a computer). This process is done by an analog to digital converter (ADC).

The ADCs used in this lab require a voltage range in which an analog value will be found. This range is known as the voltage references of the ADC and is dependent of the type of microcontroller (information found in a data sheet). In the case of MSP432 the minimum voltage that can be converted is 0 V. This is referred to as V_{REF-}, thus in the case of MSP432 $V_{REF-} = 0$ V. Similarly, the maximum value that can be converted is V_{REF+} which is 3.3 V for MSP432 ($V_{REF+} = 3.3$ V).

As mentioned before a microcontroller (and any computer) operates using a binary number system. This means that when an ADC digitizes a signal (voltage value within the voltage reference range) it must be represented on the same binary system. An analog value is represented in its digital format by dividing the voltage reference range by number of possible values the ADC can resolve. The number of values is determined by the bit size of the ADC. An ADC can resolve 2^n different values where *n* is the bit size of the ADC.

The MSP432 has a 14-bit ADC and a voltage reference from $V_{REF-} = 0$ V to $V_{REF+} = 3.3$ V. Thus the ADC can represent 2^{14} values or values. By mapping the values 0 to 16384 from 0 to 3.3 V, the MSP432 ADC has a resolution of 3.3 V / 16384 units or 0.0002 V per division. Formally the resolution of an ADC can be represented with the following equation:

$$Resolution = \frac{1}{2^n}\left(V_{REF+} - V_{REF-}\right)$$

The higher the bit size of the ADC, the better the resolution becomes and the more accurate the digitized reading becomes. However, an ADC with a higher bit size is also more susceptible to noise, more expensive and slower. The bit size of an ADC is an important consideration when choosing a microcontroller and all aspects of the ADC must be taken into account in order to pick the best choice for a given application.

When digitizing a signal, the time at which samples are taken as well as the amplitude are converted to discrete values. This means that it is important to consider at what speed samples will be taken. For example, in figure 2.4.1 above, a sine wave can be sampled at discrete time points to yield the image on the right. However, if the sampling rate were to decrease to 0.25 seconds, there would only be five samples taken. In this case, only the five samples at 0, 0.25, 0.5, 0.75 and 1 seconds would be taken. There are a multitude of different signals that could match this "down sampled" version of the signal and a computer would have no way to distinguish which to use. This concept is illustrated in figure 2.4.2 below.

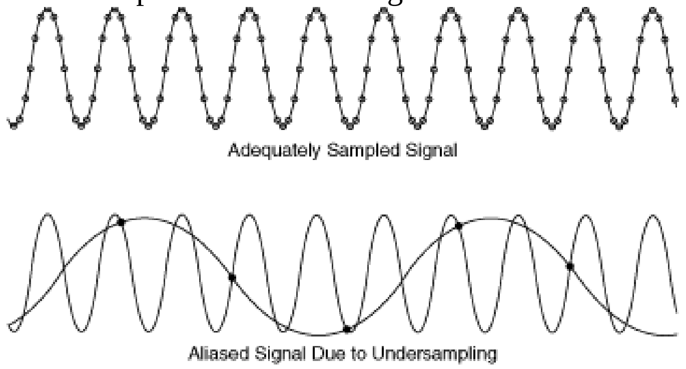

Adequately Sampled Signal
Aliased Signal Due to Undersampling

Figure 2.4.2: An analog signal sampled adequately is shown above. The down sampled (less frequent sampling) is shown below with a secondary wave (not the original signal) that could also be extrapolated form the results.

As shown above, if a signal is not sampled fast enough, it can be aliased (look like another signal). To avoid this, the Shannon-Nyquist sampling theorem is used. The Shannon-Nyquist sampling theorem states that in order reconstruct a signal from samples taken, the sampling rate must be two times that of the highest frequency component in the signal. For example, say a sine way with frequency 40 Hz is being sampled. This 40 Hz wave must be sampled at a rate of <u>at least</u> 80 Hz (40 Hz * 2 = 80 Hz). Often times a sampling rate of about five times the highest frequency component of a signal is used.

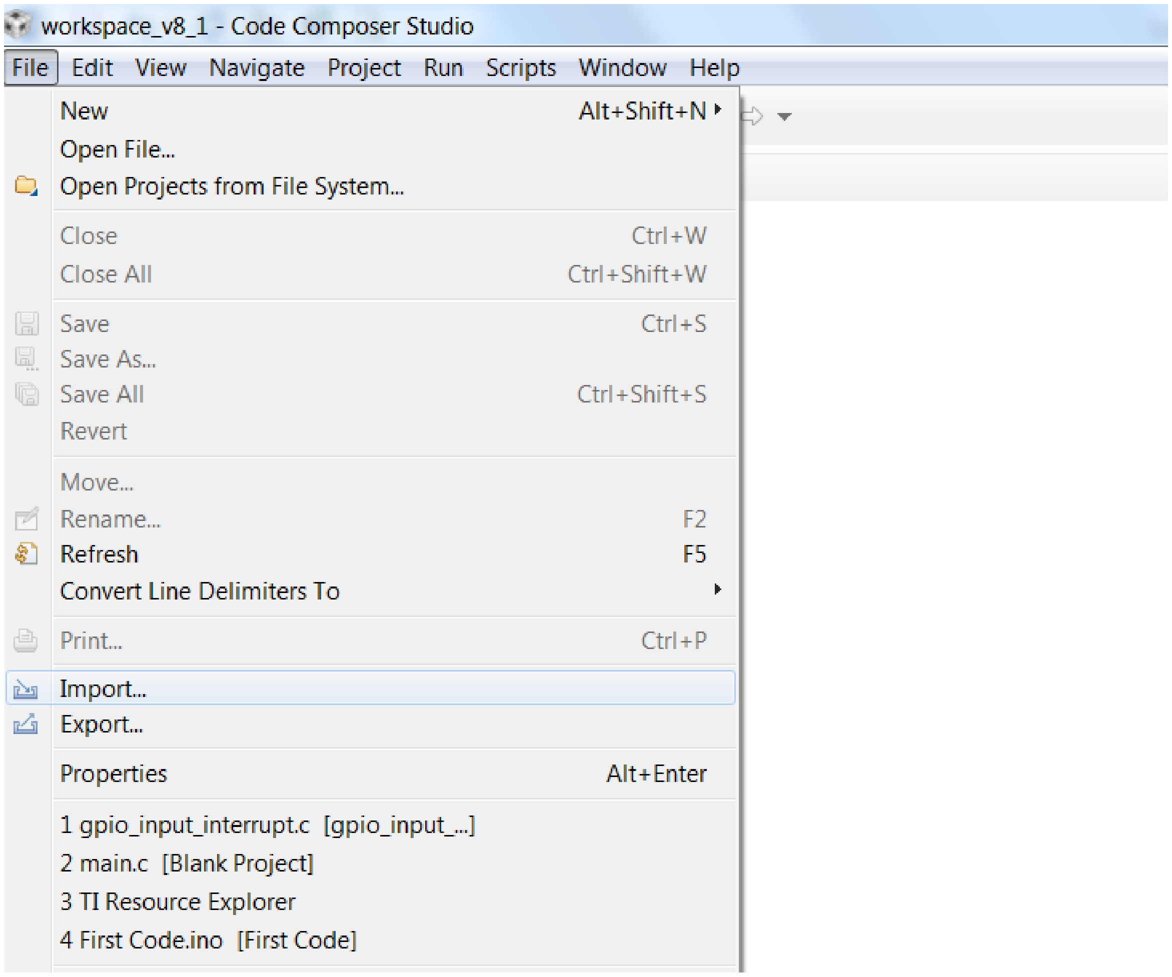
Purpose:

A microcontroller must convert the analog signals measured from the environment in to usable digital data via the use of and ADC. This section will focus on utilizing the MSP432's ADC to read in data from the environment and illustrate the benefits and shortfalls of this particular hardware.

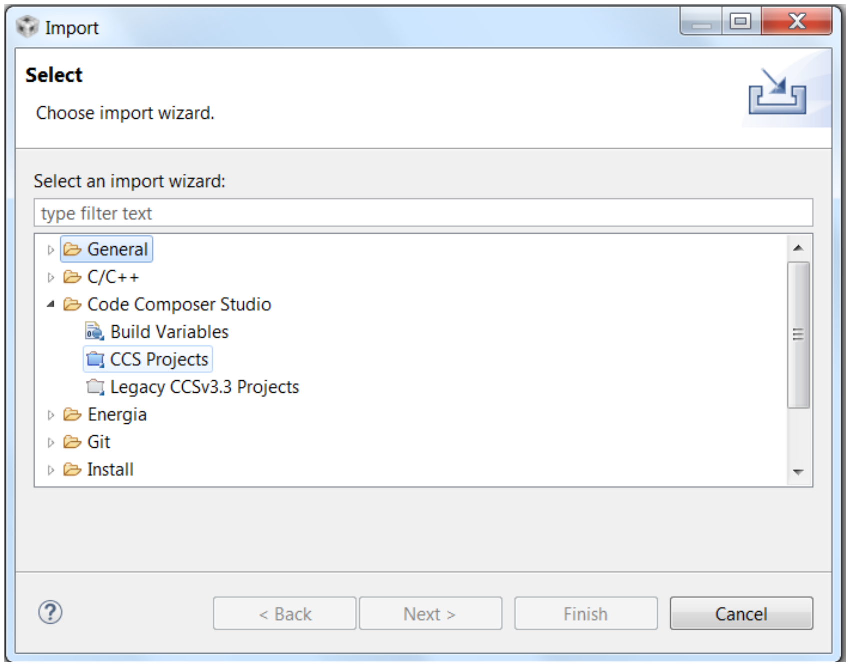
Terms:

- ADC (Analog to Digital Converter): A device that converts continues signals into discrete signals.
- ADC Resolution: The level to which the ADC can distinguish one analog input from another. The resolution is a function of the number of parts the maximum signal value can be divided. Usually measured in volts.
- Voltage Reference Range: The range in which the ADC can convert.

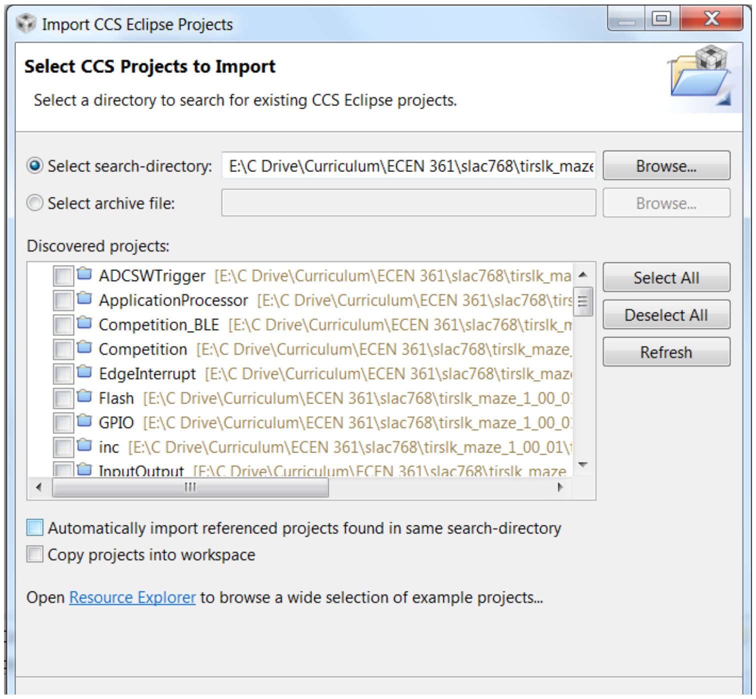
Let's look at a specific example. Let's use some example code. First, download the slac768.zip file. Unzip this to a directory. Now you can import a basic A/D project. To do this in code Composer first click on the Import selection:



When the dialog box pops up select the CCS Projects selection:



Then Click Next. You now need to select the directory where you unzip the slac768.zip file:


Import CCS Eclipse Projects
Select CCS Projects to Import
Select a directory to search for existing CCS Eclipse projects.
Browse...
Discovered projects:
Select All
Deselect All
Refresh
Open Resource Explorer to browse a wide selection of example projects...

Select search-directory: E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze

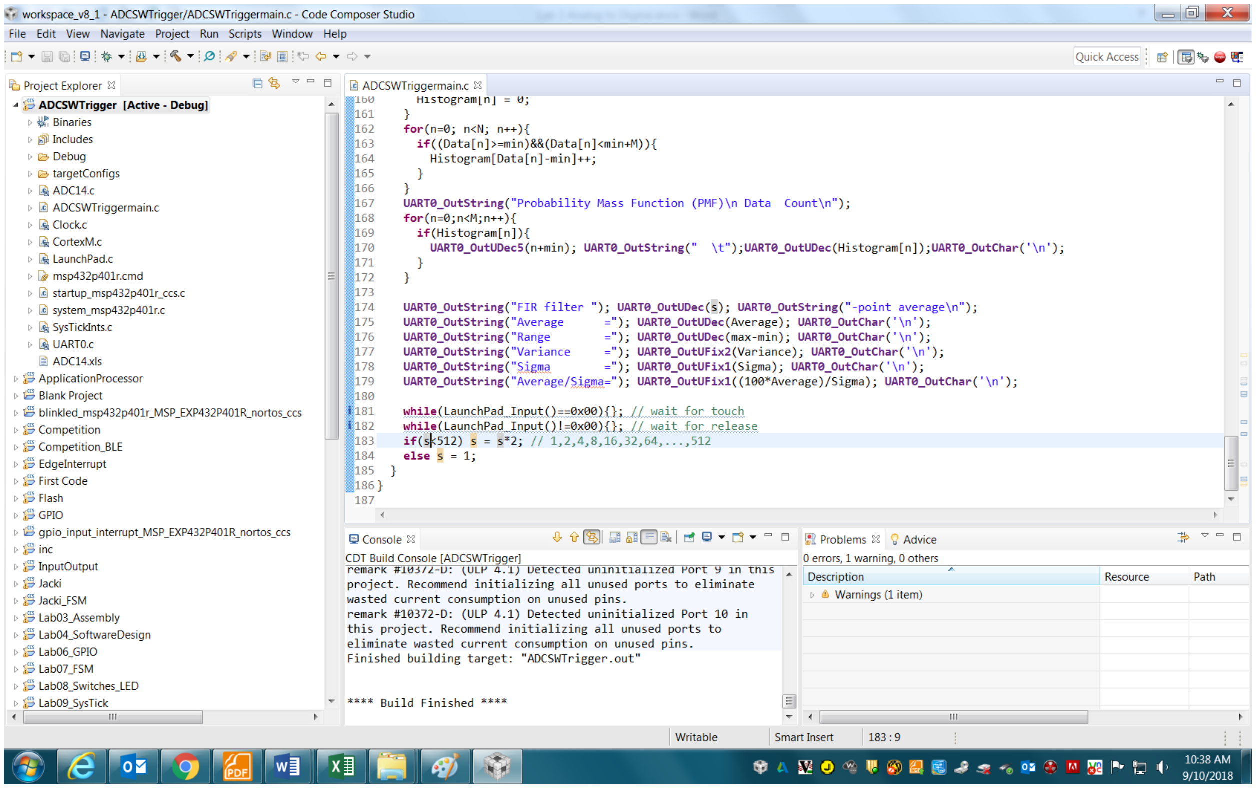
Select archive file:

ADCSWTrigger [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_ma
ApplicationProcessor [E:\C Drive\Curriculum\ECEN 361\slac768\tirs
Competition_BLE [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_m
Competition [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze
EdgeInterrupt [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maz
Flash [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze_1_00_0
GPIO [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze_1_00_0
inc [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze_1_00_01\
InputOutput [E:\C Drive\Curriculum\ECEN 361\slac768\tirslk_maze

Automatically import referenced projects found in same search-directory
Copy projects into workspace

In order to get all the code you need, go ahead and click Select All and also select Copy project into workspace (this will copy all the relevant files into your workspace.)

Now click Finish. The project should now be shown in your IDE project window. Go ahead and click on ADCSWTrigger and expand the directory:



```
    Histogram[n] = 0;
  }
  for(n=0; n<N; n++){
    if((Data[n]>=min)&&(Data[n]<min+M)){
      Histogram[Data[n]-min]++;
    }
  }
  UART0_OutString("Probability Mass Function (PMF)\n Data  Count\n");
  for(n=0;n<M;n++){
    if(Histogram[n]){
      UART0_OutUDec5(n+min); UART0_OutString("  \t");UART0_OutUDec(Histogram[n]);UART0_OutChar('\n');
    }
  }

  UART0_OutString("FIR filter "); UART0_OutUDec(s); UART0_OutString("-point average\n");
  UART0_OutString("Average      ="); UART0_OutUDec(Average); UART0_OutChar('\n');
  UART0_OutString("Range        ="); UART0_OutUDec(max-min); UART0_OutChar('\n');
  UART0_OutString("Variance     ="); UART0_OutUFix2(Variance); UART0_OutChar('\n');
  UART0_OutString("Sigma        ="); UART0_OutUFix1(Sigma); UART0_OutChar('\n');
  UART0_OutString("Average/Sigma="); UART0_OutUFix1((100*Average)/Sigma); UART0_OutChar('\n');

  while(LaunchPad_Input()==0x00){}; // wait for touch
  while(LaunchPad_Input()!=0x00){}; // wait for release
  if(s<512) s = s*2; // 1,2,4,8,16,32,64,...,512
  else s = 1;
 }
}
```

CDT Build Console [ADCSWTrigger]

```
remark #10372-D: (ULP 4.1) Detected uninitialized Port 9 in this
project. Recommend initializing all unused ports to eliminate
wasted current consumption on unused pins.
remark #10372-D: (ULP 4.1) Detected uninitialized Port 10 in
this project. Recommend initializing all unused ports to
eliminate wasted current consumption on unused pins.
Finished building target: "ADCSWTrigger.out"


**** Build Finished ****
```

0 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		

There are a number of different .c files. To understand what is going on first select the ADCSWTriggermain.c file. Then go to the main() function and let’s look at the code:

```
int main(void){ int32_t n; uint32_t min,max,s=32;
```

The first statements configure the MSP432. You’ll set the clock to 48 MHz, setup Analog input 6, connected to pin 4.7, set up LPF filter on the data, configure a UART so you can send out debug data, and initialize the MSP432. You’re also going to set up a Watchdog timer to 1000 Hz.

```
  Clock_Init48MHz();
  ADCflag = 0;
  ADC0_InitSWTriggerCh6();   // initialize ADC sample P4.7/A6
  LPF_Init(ADC_In6(),s);
  UART0_Init();              // initialize UART0 115,200 baud rate
  LaunchPad_Init();
  SysTick_Init(48000,2);     // 1000 Hz sampling
```

Now you’ll output a debug string and enable interrupts

```
  UART0_OutString("ADCSWTrigger project\nValvano July 2017\n1000 Hz 14-bit
sampling\nConnect analog signal to P4.7\n");
  EnableInterrupts();
```

Now here is a while forever loop

```
  while(1){
```

Get a set of data samples

```
     UART0_OutString("\nADC resolution test\n");
     LPF_Init(ADC_In6(),s);
     for(n=0; n<2*s; n++){
      while(ADCflag == 0){};
      ADCflag = 0; // skip the first 2*s
    }
```

Transfer the data to the array Data[n]

```
    UART0_OutString("Collect "); UART0_OutUDec(N); UART0_OutString(" samples\n");
    Sum = Sum2 = 0;
    for(n=0; n<N; n++){
      while(ADCflag == 0){};
      Sum = Sum+ADCvalue;            // 14bits*100 = 17 bits
      Data[n] = ADCvalue;
      ADCflag = 0;
    }
    Average = Sum/N;
```

Now calculate the data characteristics

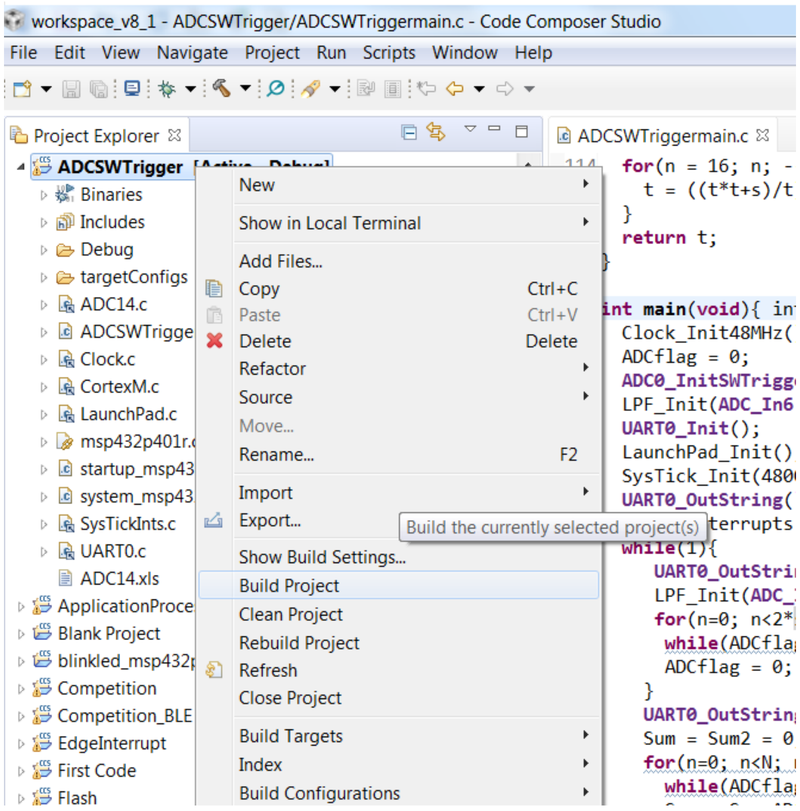
```
    for(n=0; n<N; n++){
      Sum2 = Sum2+(Data[n]-Average)*(Data[n]-Average); // 28bits*100 = 31 bits
    }
    Variance = (100*Sum2)/(N-1);
    Sigma = sqrt(Variance);

    min = 16384; max = 0;
    for(n=0; n<N; n++){
      if(Data[n] < min)
        min = Data[n]; // smallest
      if(Data[n] > max)
        max = Data[n]; // largest
      }
    for(n=0;n<M;n++){
      Histogram[n] = 0;
    }
    for(n=0; n<N; n++){
      if((Data[n]>=min)&&(Data[n]<min+M)){
        Histogram[Data[n]-min]++;
      }
    }
    UART0_OutString("Probability Mass Function (PMF)\n Data  Count\n");
    for(n=0;n<M;n++){
      if(Histogram[n]){
        UART0_OutUDec5(n+min); UART0_OutString("
\t");UART0_OutUDec(Histogram[n]);UART0_OutChar('\n');
```

```
      }
    }

    UART0_OutString("FIR filter "); UART0_OutUDec(s); UART0_OutString("-point
average\n");
    UART0_OutString("Average      ="); UART0_OutUDec(Average); UART0_OutChar('\n');
    UART0_OutString("Range        ="); UART0_OutUDec(max-min); UART0_OutChar('\n');
    UART0_OutString("Variance     ="); UART0_OutUFix2(Variance); UART0_OutChar('\n');
    UART0_OutString("Sigma        ="); UART0_OutUFix1(Sigma); UART0_OutChar('\n');
    UART0_OutString("Average/Sigma="); UART0_OutUFix1((100*Average)/Sigma);
UART0_OutChar('\n');
```

Pause here until the user hits and releases the SW1 Button

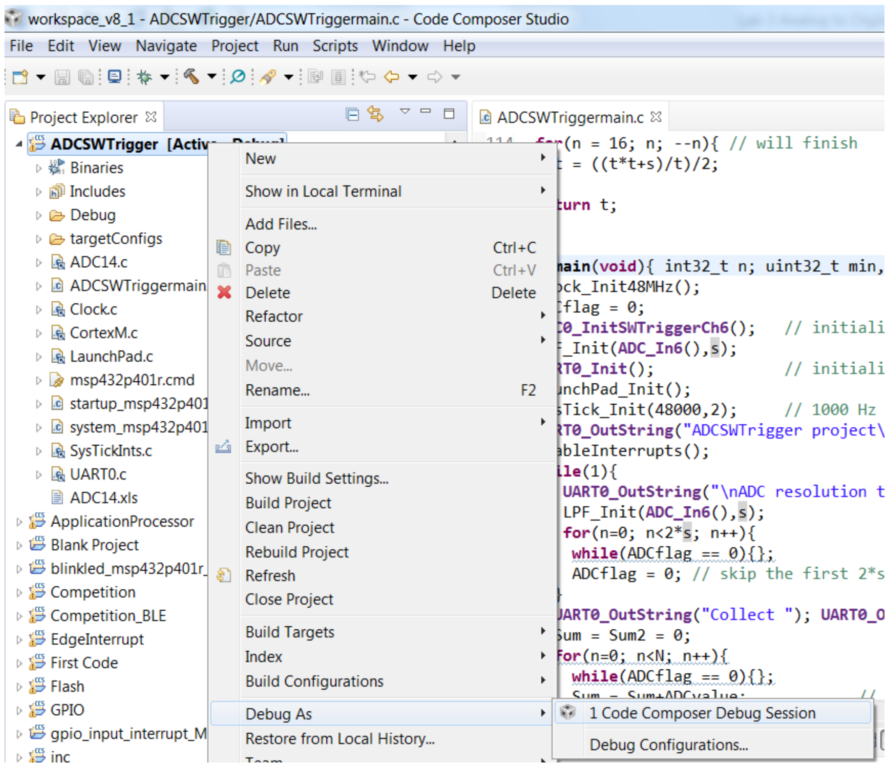
```
    while(LaunchPad_Input()==0x00){}; // wait for touch
    while(LaunchPad_Input()!=0x00){}; // wait for release
    if(s<512) s = s*2; // 1,2,4,8,16,32,64,...,512
    else s = 1;
  }
}
```

To run this code you'll first need to Build the Project.



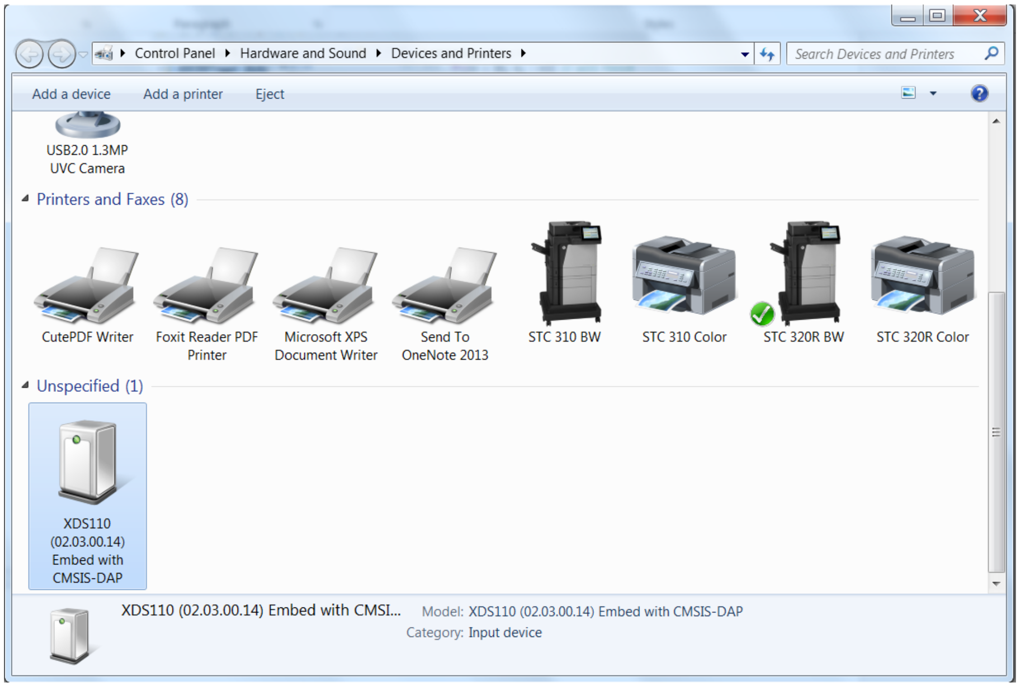
```
  for(n = 16; n; -
    t = ((t*t+s)/t
  }
  return t;
}

int main(void){ in
  Clock_Init48MHz(
  ADCflag = 0;
  ADC0_InitSWTrigg
  LPF_Init(ADC_In6
  UART0_Init();
  LaunchPad_Init()
  SysTick_Init(480
  UART0_OutString(
              terrupts
  while(1){
    UART0_OutStri
    LPF_Init(ADC_
    for(n=0; n<2*
      while(ADCfla
      ADCflag = 0;
    }
    UART0_OutStrin
    Sum = Sum2 = 0
    for(n=0; n<N;
      while(ADCfla
```

Now you can Run the Project by selecting

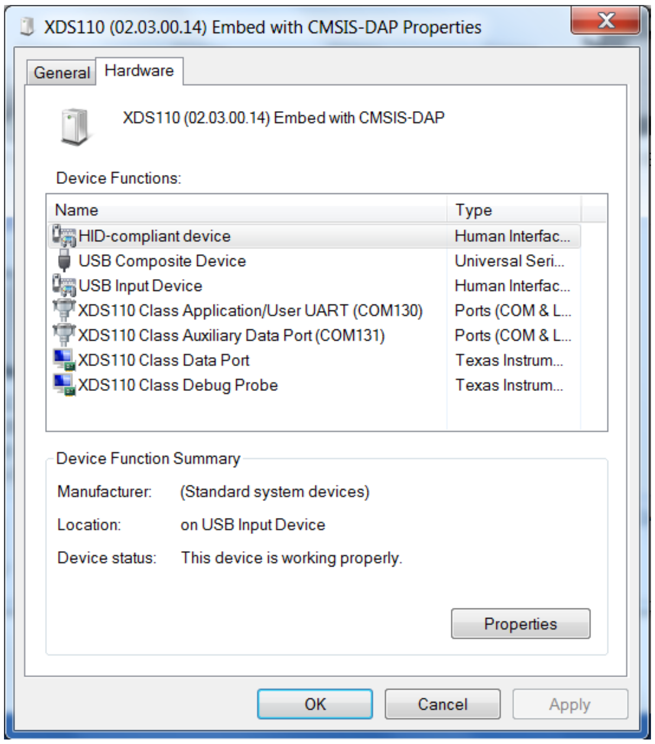


```
  for(n = 16; n; --n){ // will finish
t = ((t*t+s)/t)/2;

turn t;

main(void){ int32_t n; uint32_t min,
ock_Init48MHz();
Cflag = 0;
C0_InitSWTriggerCh6();   // initiali
_Init(ADC_In6(),s);
RT0_Init();              // initiali
unchPad_Init();
sTick_Init(48000,2);     // 1000 Hz
RT0_OutString("ADCSWTrigger project\
ableInterrupts();
ile(1){
 UART0_OutString("\nADC resolution t
 LPF_Init(ADC_In6(),s);
 for(n=0; n<2*s; n++){
  while(ADCflag == 0){};
  ADCflag = 0; // skip the first 2*s
}
UART0_OutString("Collect "); UART0_O
Sum = Sum2 = 0;
for(n=0; n<N; n++){
 while(ADCflag == 0){};
 Sum = Sum+ADCvalue;         //
```

The debug view should appear. Now you will want to open a terminal window to communicate with the application. To do this first find out the terminal COM port using the Control Panel->Devices and Printers.


Control Panel ▸ Hardware and Sound ▸ Devices and Printers ▸
Search Devices and Printers
Add a device
Add a printer
Eject
USB2.0 1.3MP UVC Camera
Printers and Faxes (8)
CutePDF Writer
Foxit Reader PDF Printer
Microsoft XPS Document Writer
Send To OneNote 2013
STC 310 BW
STC 310 Color
STC 320R BW
STC 320R Color
Unspecified (1)
XDS110 (02.03.00.14) Embed with CMSIS-DAP
XDS110 (02.03.00.14) Embed with CMSI...
Model: XDS110 (02.03.00.14) Embed with CMSIS-DAP
Category: Input device

Double Click on the XDS110 selection, and then select the Hardware tab:


XDS110 (02.03.00.14) Embed with CMSIS-DAP Properties

General | Hardware

XDS110 (02.03.00.14) Embed with CMSIS-DAP

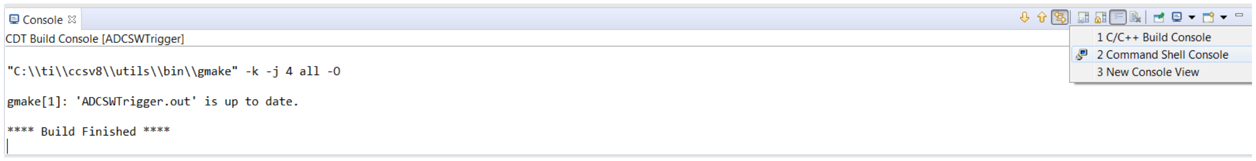
Device Functions:

Name	Type
HID-compliant device	Human Interfac...
USB Composite Device	Universal Seri...
USB Input Device	Human Interfac...
XDS110 Class Application/User UART (COM130)	Ports (COM & L...
XDS110 Class Auxiliary Data Port (COM131)	Ports (COM & L...
XDS110 Class Data Port	Texas Instrum...
XDS110 Class Debug Probe	Texas Instrum...

Device Function Summary

Manufacturer:	(Standard system devices)
Location:	on USB Input Device
Device status:	This device is working properly.

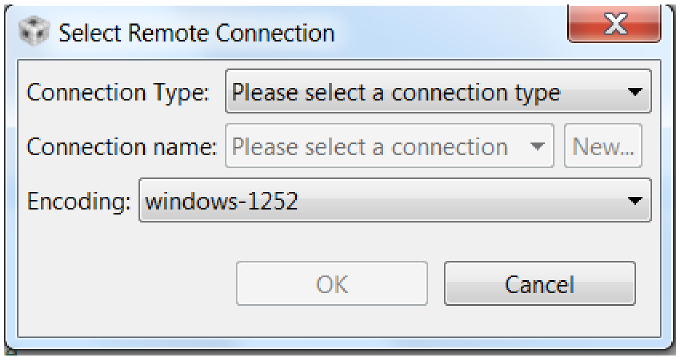
Note the Port number of the Application/User UART (in this case COM130). Now go back to the IDE where you are going to open a terminal window. To do this select the add terminal selection and select the Command Shell Console:



CDT Build Console [ADCSWTrigger]

```
"C:\\ti\\ccsv8\\utils\\bin\\gmake" -k -j 4 all -O

gmake[1]: 'ADCSWTrigger.out' is up to date.

**** Build Finished ****
```

The following dialogue box will open:

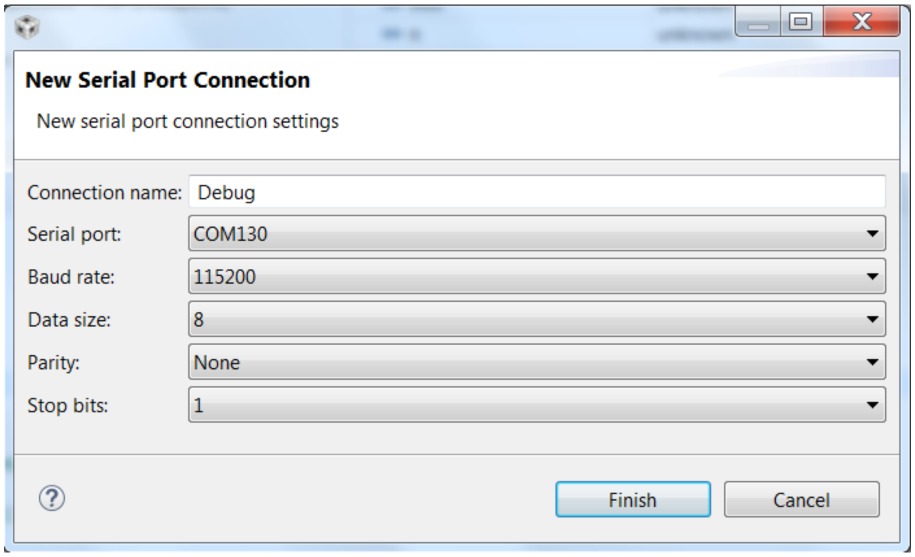

Select Remote Connection

Connection Type: Please select a connection type

Connection name: Please select a connection

Encoding: windows-1252

The Connection Type should be set to Serial Port. Select the New… next to Connection name, then fill the values out

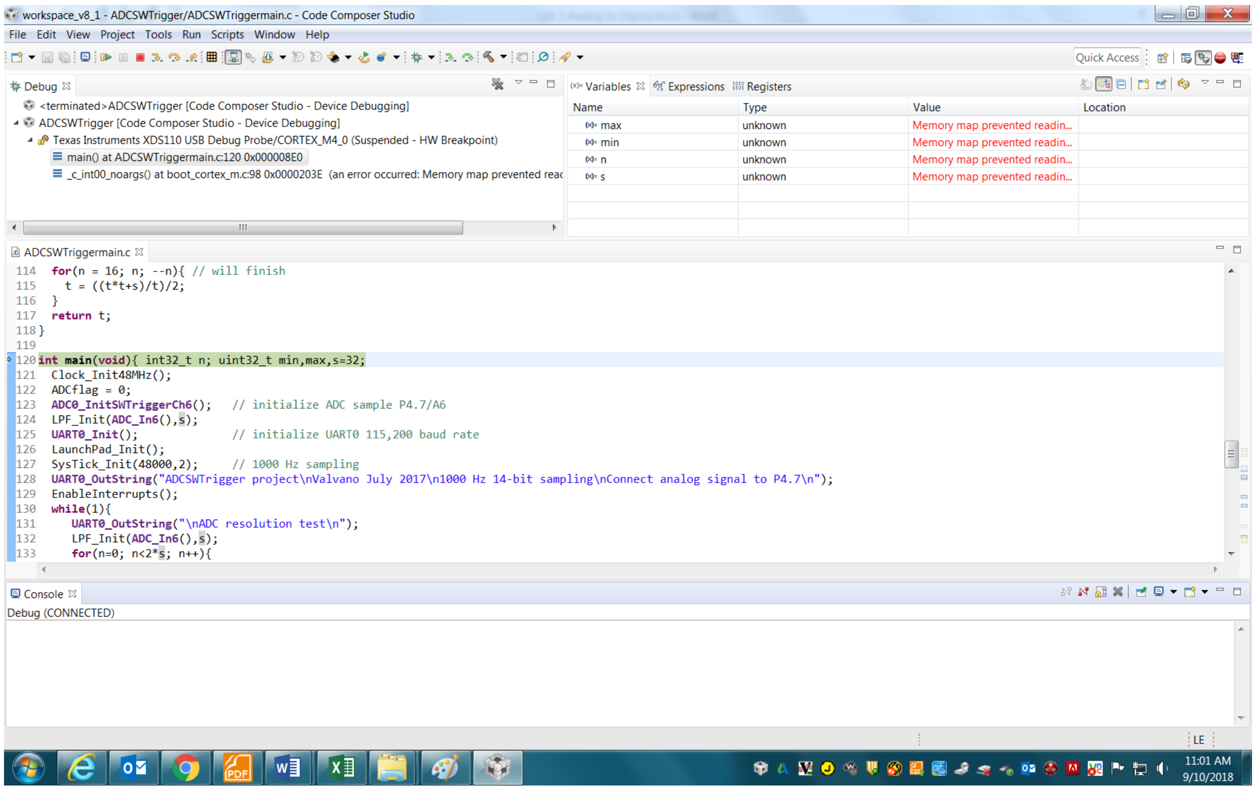


New Serial Port Connection

New serial port connection settings

Connection name:	Debug
Serial port:	COM130
Baud rate:	115200
Data size:	8
Parity:	None
Stop bits:	1

The Serial port number will be the one you found just above. Now select Finish. Then Click OK. Your system should now look like this:



workspace_v8_1 - ADCSWTrigger/ADCSWTriggermain.c - Code Composer Studio

```
<terminated>ADCSWTrigger [Code Composer Studio - Device Debugging]
ADCSWTrigger [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspended - HW Breakpoint)
    main() at ADCSWTriggermain.c:120 0x000008E0
    _c_int00_noargs() at boot_cortex_m.c:98 0x0000203E  (an error occurred: Memory map prevented reac
```

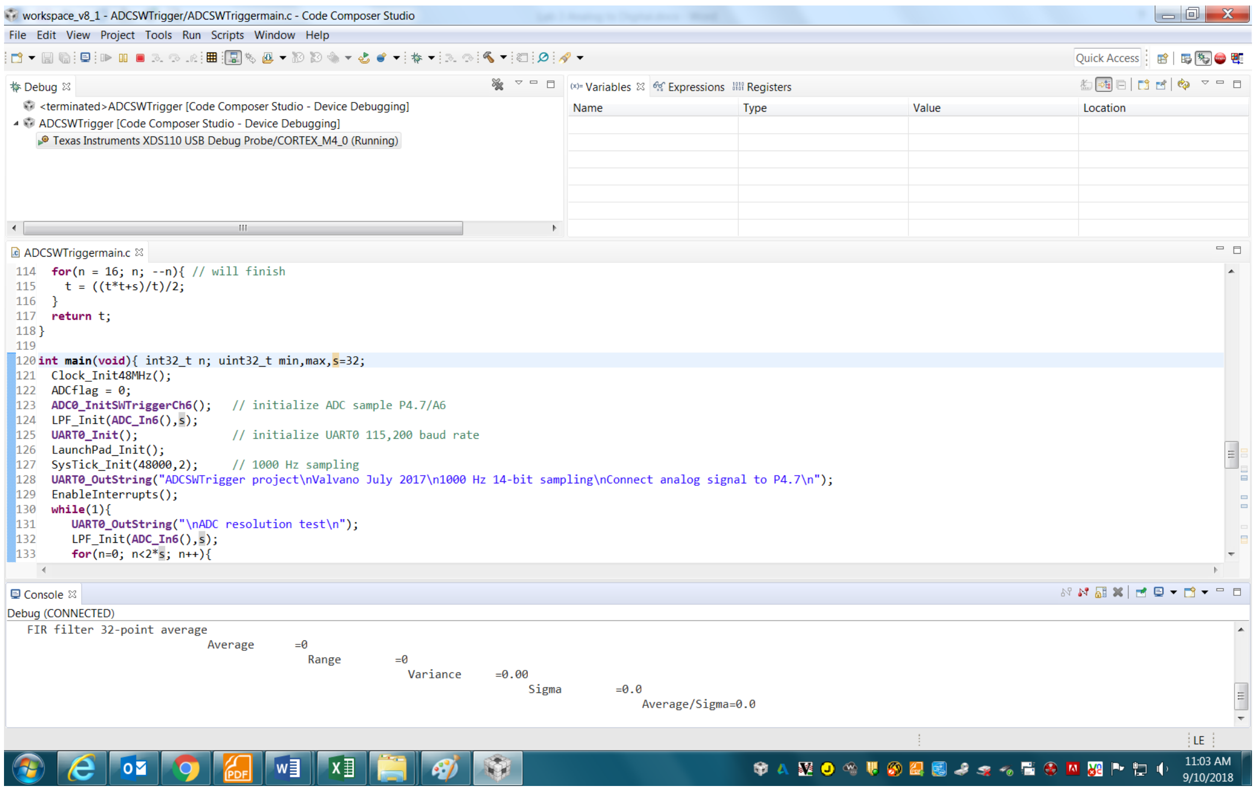
Name	Type	Value	Location
max	unknown	Memory map prevented readin...	
min	unknown	Memory map prevented readin...	
n	unknown	Memory map prevented readin...	
s	unknown	Memory map prevented readin...	

ADCSWTriggermain.c

```
  for(n = 16; n; --n){ // will finish
    t = ((t*t+s)/t)/2;
  }
  return t;
}

int main(void){ int32_t n; uint32_t min,max,s=32;
  Clock_Init48MHz();
  ADCflag = 0;
  ADC0_InitSWTriggerCh6();   // initialize ADC sample P4.7/A6
  LPF_Init(ADC_In6(),s);
  UART0_Init();              // initialize UART0 115,200 baud rate
  LaunchPad_Init();
  SysTick_Init(48000,2);     // 1000 Hz sampling
  UART0_OutString("ADCSWTrigger project\nValvano July 2017\n1000 Hz 14-bit sampling\nConnect analog signal to P4.7\n");
  EnableInterrupts();
  while(1){
     UART0_OutString("\nADC resolution test\n");
     LPF_Init(ADC_In6(),s);
     for(n=0; n<2*s; n++){
```

```
Debug (CONNECTED)
```

The Debug window is at the bottom. You code is paused at the main() function. To execute the code select Run->Resume. Now you should see this:


workspace_v8_1 - ADCSWTrigger/ADCSWTriggermain.c - Code Composer Studio

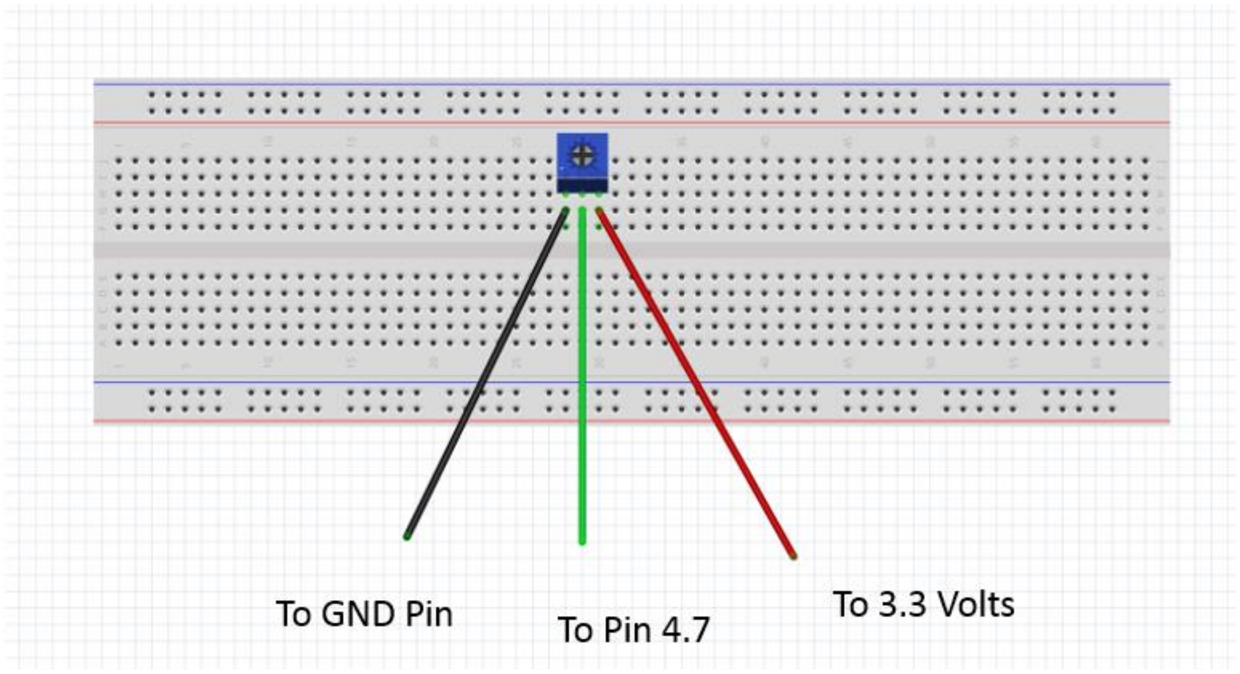
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
<terminated>ADCSWTrigger [Code Composer Studio - Device Debugging]
ADCSWTrigger [Code Composer Studio - Device Debugging]
Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Running)
Variables
Expressions
Registers
Name
Type
Value
Location
ADCSWTriggermain.c
Console
LE
11:03 AM
9/10/2018

```c
  for(n = 16; n; --n){ // will finish
    t = ((t*t+s)/t)/2;
  }
  return t;
}

int main(void){ int32_t n; uint32_t min,max,s=32;
  Clock_Init48MHz();
  ADCflag = 0;
  ADC0_InitSWTriggerCh6();   // initialize ADC sample P4.7/A6
  LPF_Init(ADC_In6(),s);
  UART0_Init();              // initialize UART0 115,200 baud rate
  LaunchPad_Init();
  SysTick_Init(48000,2);     // 1000 Hz sampling
  UART0_OutString("ADCSWTrigger project\nValvano July 2017\n1000 Hz 14-bit sampling\nConnect analog signal to P4.7\n");
  EnableInterrupts();
  while(1){
     UART0_OutString("\nADC resolution test\n");
     LPF_Init(ADC_In6(),s);
     for(n=0; n<2*s; n++){
```

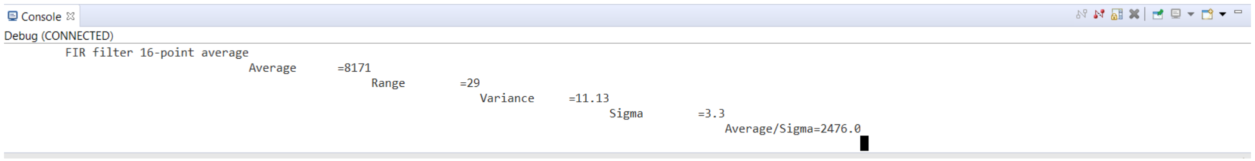
Debug (CONNECTED)

```
   FIR filter 32-point average
                              Average      =0
                                             Range        =0
                                                            Variance     =0.00
                                                                             Sigma        =0.0
                                                                                             Average/Sigma=0.0
```

The system takes in values, then computes the Average, Range, Variance, Sigma, and Average/Sigma. This will repeat itself every time you press switch S1. Since no signal is connected these are all 0. So now let’s connect a signal. To do this you’ll want to create some sort of voltage on Pin 4.7 between 0 and 3.3 volts. To do this you use a signal source, or you can use a potentiometer. Here is a very simple potentiometer circuit:

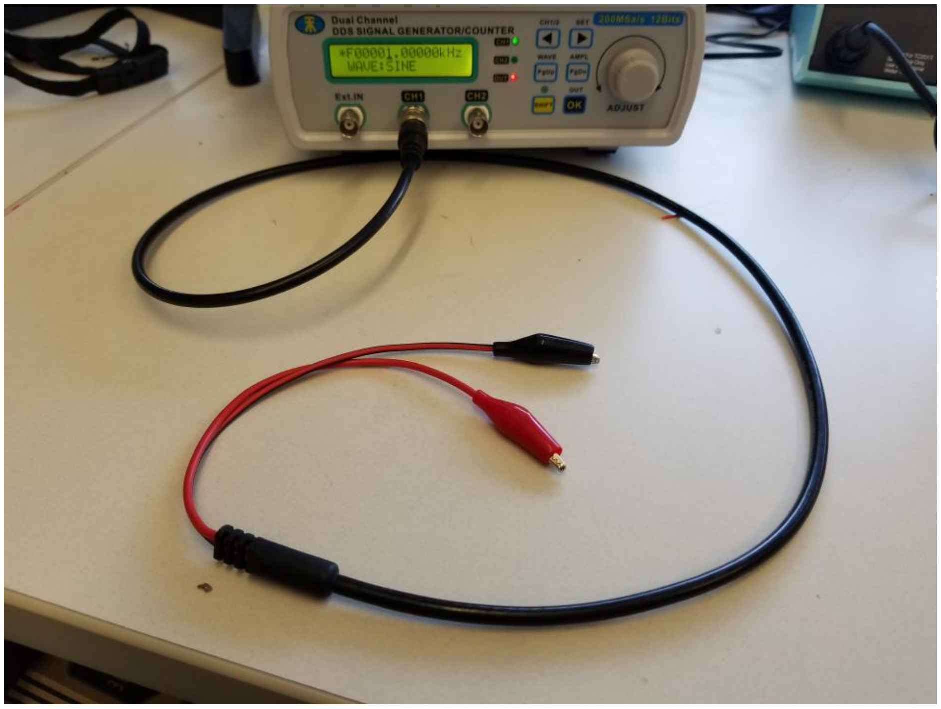


Here are some results from a DC offset signal like this:



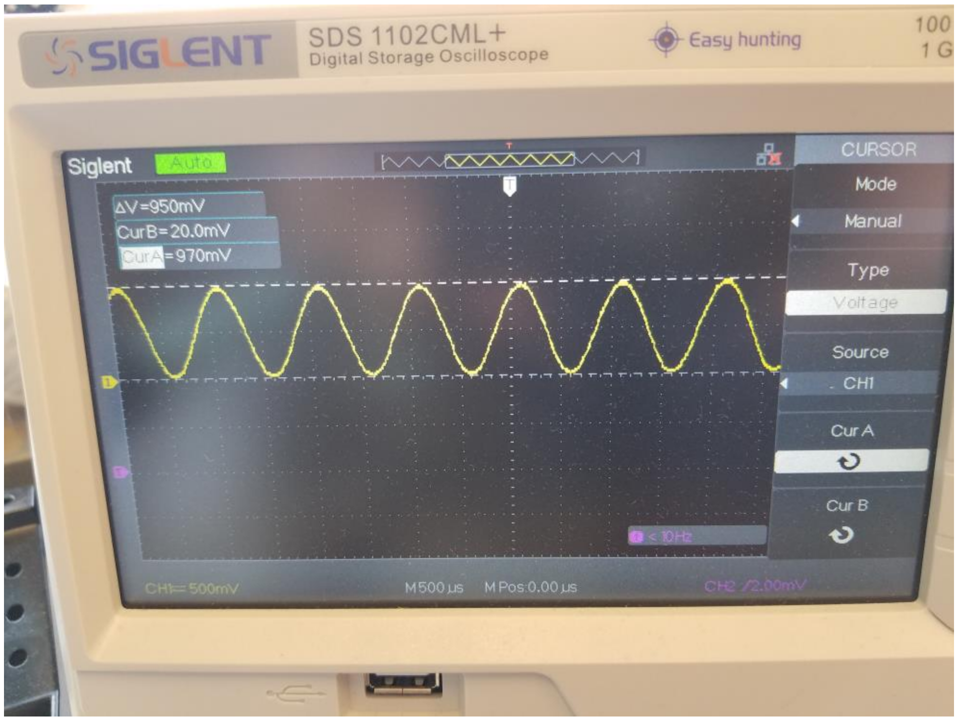
```
        FIR filter 16-point average
                                Average      =8171
                                                Range        =29
                                                              Variance      =11.13
                                                                              Sigma        =3.3
                                                                                 Average/Sigma=2476.0
```

To connect to a signal source use a BNC cable with alligator clips on the end, like this:

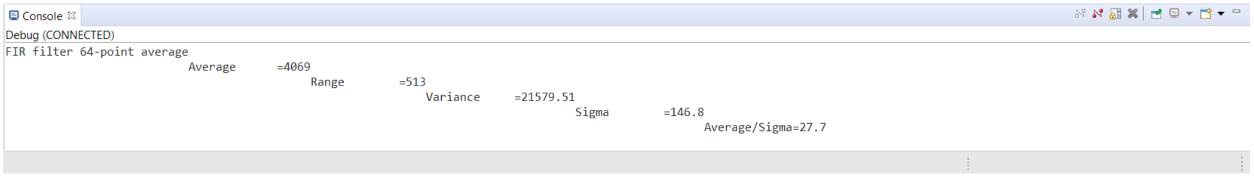


Make sure you don’t exceed 3.3 volts or you will damage your MSP432. Here are the results if you set up the following waveform:

1 Volt Pk-Pk
1 kHz Frequency
Amplitude Offset: 100%
The signal looks like this:



Here are the results:


```
Console
Debug (CONNECTED)
FIR filter 64-point average
                           Average       =4069
                                                 Range         =513
                                                                 Variance      =21579.51
                                                                                       Sigma         =146.8
                                                                                                       Average/Sigma=27.7
```

Lab results:

1) Capture the screen with some valid measurements using the code and the potentiometer circuit. Share what the input voltage looks like and what your measurements say.

2) Capture the screen with some valid measurements using the code and a function generator. Share the input signal and why the data is the way it is.

3) Answer these questions:

   a) What signal frequency would violate this systems Nyquist (sampling) Frequency?
   b) What is the voltage resolution of this system?
   c) What is the sampling inaccuracy of this system?