**Design Assignment 3:**

**Signal Conversion**

****

**Embedded Systems**

**ELC 411**

**Matthew Strickland and Jacob Levine**

**Submission: 10/26/17**

1. Equation to estimate ADC input voltage

ADC\_input\_voltage = sar\_count truncated to the millivolt

??? – no

ADC\_input\_voltage\_est = sar\_count \* 2.048/4096;

1. Debugged code

#include "project.h"  
#include "stdio.h"  
  
const double maxVo = 2.048; // student added constant for calculations  
const uint8\_t maxCounts = 255; // student added constant for calculations  
  
CY\_ISR\_PROTO(my\_isr);  
  
CY\_ISR(my\_isr)  
{  
 static int count = 0; // must be static, so value retained between interrupts, one time initialization  
   
 //code here to alternately drive VDAC8 to 1/4 full scale and 3/4 full scale  
   
   
 if (count == 0)  
 VDAC8\_SetValue(maxCounts / 4);  
   
 if (count == 1)  
 VDAC8\_SetValue(maxCounts \* 3 / 4);   
 // Don't change code below  
 isr\_1\_ClearPending();  
 count = 1 - count; // toggle between 0 and 1  
}  
  
int  
main( void )  
  
{  
 int sar\_result; // holds result from ADC\_SAR  
 int volts\_int; // integer part of volts, for string formatting to overcome bug in PSoC Creator  
 int volts\_frac; // fractional part of volts, for string formatting to overcome bug in PSoC Creator  
 char disp\_str[17]; // char array large enough to hold one line for display  
   
 double volts; // student computes this, as a function of sar\_result  
   
 CyGlobalIntEnable; /\* Enable global interrupts. \*/  
   
 // Initialization, start your engines ...  
 ADC\_SAR\_Start();  
 ADC\_SAR\_StartConvert(); // Needed because ADC\_SAR is free-running  
 VDAC8\_Start();  
 isr\_1\_StartEx(my\_isr);  
 LCD\_Char\_Start();  
   
 // Loop forever  
 while (SW3\_Read() == 1)  
 {  
 sar\_result = ADC\_SAR\_GetResult16(); // get new ADC value  
   
 //code below, to convert sar\_result to the corresponding (floating point) voltage  
   
 volts = ADC\_SAR\_CountsTo\_Volts(sar\_result);

\*\*\* But what is this function???

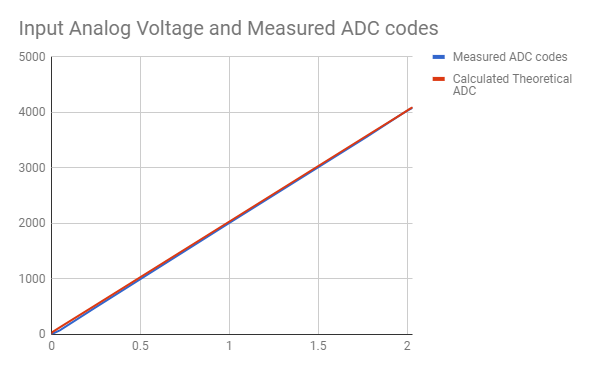
I will give you credit for using PSoC function in code.

// Don't change anything below!  
 volts\_int = (int) volts; // get integer part  
 volts\_frac = (int) ((volts - volts\_int) \* 1000 + 0.5); // get fractional part as a 3-digit integer  
   
 // Display the string on top line, left justified  
 LCD\_Char\_ClearDisplay();  
 LCD\_Char\_Position(0, 0);  
 sprintf( disp\_str, "code=%4d", sar\_result);  
 LCD\_Char\_PrintString(disp\_str);  
  
 // Display the string on bottom line, left justified  
 LCD\_Char\_Position(1, 0);  
 sprintf( disp\_str, "volts=%d.%03d", volts\_int, volts\_frac );  
 LCD\_Char\_PrintString(disp\_str);  
   
 CyDelay(500); // a little time for display to stabilize  
 }  
}

1. Table of measured values

|  |  |  |
| --- | --- | --- |
| Input voltage setting | Input voltage meter reading | Measured ADC codes (approximate average) |
| 0 mV | 0 | 0 |
| 1 mV | 0 | 0 |
| 4 mV | 0 | 0 |
| 16 mV | 0 | 5 |
| 64 mV | .0467 | 75 |
| 256 mV | .2392 | 470 |
| 1024 mV | 1.0072 | 2025 |
| 1.792 V | 1.7750 | 3565 |
| 1.982 V | 1.9674 | 3965 |
| 2.032 V | 2.0148 | 4055 |
| 2.044 V | 2.0279 | 4078 |
| 2.047 V | 2.031 | 4083 |
| 2.048 V | 2.031 | 4087 |

1. Plot of measured ADC codes vs. measured analog\_in



1. Maximum error and RMS error

Max error was 53, erms = 81.51073549

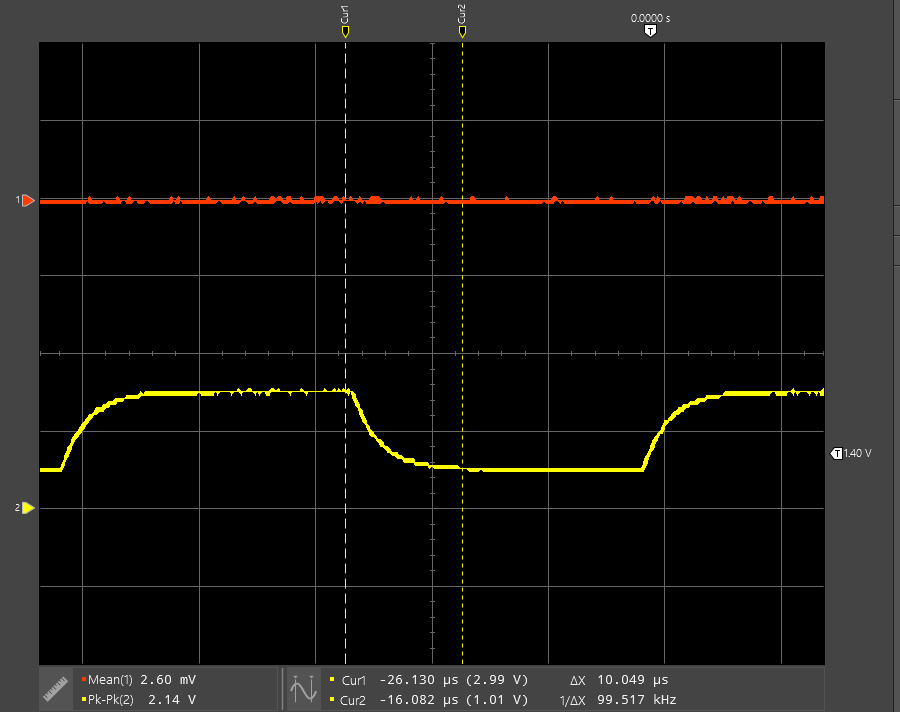
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Analog Voltage | Measured ADC codes | Calculated Theoretical ADC | squared error | error |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 4 | 2 |
| 0 | 0 | 8 | 64 | 8 |
| 0 | 5 | 32 | 729 | 27 |
| 0.0467 | 75 | 128 | 2809 | 53 |
| 0.2392 | 470 | 512 | 1764 | 42 |
| 1.0072 | 2025 | 2048 | 529 | 23 |
| 1.775 | 3565 | 3584 | 361 | 19 |
| 1.9674 | 3965 | 3964 | 1 | -1 |
| 2.0148 | 4055 | 4064 | 81 | 9 |
| 2.0279 | 4078 | 4088 | 100 | 10 |
| 2.031 | 4083 | 4094 | 121 | 11 |
| 2.031 | 4087 | 4096 | 81 | 9 |
|  |  | summed error | 6644 |  |
|  |  | erms | 81.51073549 |  |

It is not possible that RMS error is larger than maximum error!

I have realized that I left out a factor of 1/K in RMS error, where K is the number of samples, my bad, I won’t deduct for it.

1. Scope Trace:

Add caption with volts/div and time/div scale factors



1. Settling time analysis

LSB = 8mV, settling time = 10.049 µs

No – 1 LSB = (3-1)/128 = 15.625 mV

How did you get LSB = 8 mV?

Settling time from datasheet: Maximum of 3.2 µs at 4V scale with a Cload of 15 pF.

Our figure is significantly different from the datasheet specifications due to our extension of the time period measured as seen on the scope. We measured to much closer than 1 LSB from the settled voltage. We also likely had a much different capacative load than 15pF as outlined in the datasheet.

1. The noise observed in the ADC measurements are likely due to inconsistencies in the components we introduced such as small errors in the actual signal generator, as well as the noise introduced by the VDAC itself as specified in the datasheet, at 750nV/sqrt(Hz).

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Expected** | **Points** | **Pts. Available** |
| Cover sheet |  | 0.5 | 0.5 |
| Equation to estimate ADC input voltage | volts = (sar\_result / 4096.0) \* 2.048; | 0.7 | 1 |
| Debugged code | Correct equation for volts, and  VDAC8\_SetValue( 64 + 128\*count ); Fully commented and properly formatted | 1 | 1 |
| Measured values in tables | Codes for specified voltages DAC period and voltages | 2 | 2 |
| Plot of ADC codes vs. analog in | Plot | 1 | 1 |
| Max error and MSE |  | 1 | 1 |
| Scope traces |  | 0.9 | 1 |
| Settling time analysis | 1 LSB = 15.625 mV | 1.5 | 2 |
| Description of "noise" in ADC measurements |  | 0.5 | 0.5 |
| **TOTAL** |  | **9.1** | **10** |