Code:

//////////////////////

//Willard Wider

//5/29/18

//ELEC 3800

//Lab 6

//////////////////////

//P5.5 is analog input

/\* DriverLib Includes \*/

**#include** <ti/devices/msp432p4xx/driverlib/driverlib.h>

/\* Standard Includes \*/

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.h>

**#define** MAX\_VALUE 0x3FFF

/\* Statics \*/

**static** **volatile** uint16\_t curADCResult = 0x0000;

uint16\_t off\_delay = 0x0000;

uint16\_t u = 0x0000;

**int** **main**(**void**)

{

/\* Halting the Watchdog \*/

MAP\_WDT\_A\_holdTimer();

/\* Setting Flash wait state \*/

MAP\_FlashCtl\_setWaitState(FLASH\_BANK0, 1);

/\* Setting DCO to 48MHz \*/

MAP\_CS\_setDCOCenteredFrequency(CS\_DCO\_FREQUENCY\_48);

//![Single Sample Mode Configure]

/\* Initializing ADC (MCLK/1/4) \*/

MAP\_ADC14\_enableModule();

//MAP\_ADC14\_initModule(ADC\_CLOCKSOURCE\_MCLK, ADC\_PREDIVIDER\_1, ADC\_DIVIDER\_4, 0);

/\* Configuring GPIOs (5.5 A0) \*/

//now it's 4.0

MAP\_GPIO\_setAsInputPin(GPIO\_PORT\_P4,GPIO\_PIN0);//clears pull up resistors i guess

MAP\_GPIO\_setAsPeripheralModuleFunctionInputPin(GPIO\_PORT\_P4, GPIO\_PIN0, GPIO\_TERTIARY\_MODULE\_FUNCTION);

/\* Configuring ADC Memory \*/

MAP\_ADC14\_configureSingleSampleMode(ADC\_MEM0, true);

MAP\_ADC14\_configureConversionMemory(ADC\_MEM0, ADC\_VREFPOS\_AVCC\_VREFNEG\_VSS, ADC\_INPUT\_A13, false);

/\* Configuring Sample Timer \*/

MAP\_ADC14\_enableSampleTimer(ADC\_MANUAL\_ITERATION);

/\* Enabling/Toggling Conversion \*/

MAP\_ADC14\_enableConversion();

MAP\_ADC14\_toggleConversionTrigger();

//![Single Sample Mode Configure]

/\* Enabling interrupts \*/

MAP\_ADC14\_enableInterrupt(ADC\_INT0);

MAP\_Interrupt\_enableInterrupt(INT\_ADC14);

MAP\_Interrupt\_enableMaster();

//setup the output pin

//remember, setting the pin HIGH turns OFF the led,

//setting pin LOW turns ON led (current cna flow)

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P1,GPIO\_PIN0);

//NOTE: sensor has full range

**while** (1)

{

//set the led on

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

//on for delay

**for**(u = 0x0000; u < curADCResult; u++) ;

//set led off

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

//off for delay

off\_delay = (0x3FFF >> 2) - curADCResult;

**for**(u = 0x0000; u < off\_delay; u++) ;

}

}

//![Single Sample Result]

/\* ADC Interrupt Handler. This handler is called whenever there is a conversion

\* that is finished for ADC\_MEM0.

\*/

**void** **ADC14\_IRQHandler**(**void**)

{

uint64\_t status = MAP\_ADC14\_getEnabledInterruptStatus();

MAP\_ADC14\_clearInterruptFlag(status);

**if** (ADC\_INT0 & status)

{

curADCResult = MAP\_ADC14\_getResult(ADC\_MEM0);

//scale the result so that it can actually look like a digital brightness

curADCResult = curADCResult >> 2;

**if**(0)

{

**printf**("curADCResult=%x\n",curADCResult);

}

MAP\_ADC14\_toggleConversionTrigger();

}

}

//![Single Sample Result]