EE4930 Advanced Microprocessor

Section 011/021, Winter 2022/23

Professor: Kerry R. Widder, Ph.D.

Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

Laboratory 6 : “RTOS System”

Name: Jorge Jurado-Garcia

Courses: EE 4930 011

Date: 01/27/2023

# Objectives

This lab objective is to implement the MSP432 adc with the use of TI-RTOS.

# Description

I For this lab, I started by importing the “event” example from CCS to use as a base and other sample for setting up the Hardware and software interrupts structures. From there, I verified that initialize my adc and timers correctly and verified that the RTOS timer was working with my adc\_start function. I read up on the SWI/HWI from the RTOS user manual and CCS documentation and implemented the structure for both using the other example provided from CCS as a reference. Wrote each task using the example structure and added code for the SWI/HWI/clock methods. During debugging, I quickly found that I that I was not using the event structure properly and was using not using a while loop in my task subroutines causing seg fault error. Once fixing these issues the program worked.

# Conclusion

In conclusion, this lab was very interesting. I learned a lot about how the structures of an RTOS works. The main difficulties I encountered were with the HWI setup and using the binary semaphores (events) correctly. Initially I couldn’t leave the HWI, and I ended up finding out that we had to read the ADC value (not just read the IV register) to clear the interrupt; additionally, I had some problems setting up the HWI using hooks, but the CCS examples uses an implementation like all the other structures. Eventually after a bit of debugging I was able to get the lab operational without any major setbacks. After taking a review on events and how to use them with your tasks I was able to get the board to work properly.

# Attachments

## Schematic

Chart, diagram

Description automatically generated

## Main

**#include** <stdio.h>

**#include** "msp432.h"

**#include** "msoe\_lib\_all.h"

**#include** "defines.h"

**#include** <MSP432P401R\_GPIO.h>

//Preprocessors

**#define** LAB6\_ON TRUE

**#define** CLEAR 21

**#if** LAB6\_ON

//Included libraries

**#include** "EE4930\_LAB6.h"

/\* XDC module Headers \*/

**#include** <xdc/std.h>

**#include** <xdc/runtime/Error.h>

**#include** <xdc/runtime/System.h>

**#include** <xdc/runtime/Timestamp.h>

/\* BIOS module Headers \*/

**#include** <ti/sysbios/BIOS.h>

**#include** <ti/sysbios/knl/Clock.h>

**#include** <ti/sysbios/knl/Swi.h>

**#include** <ti/sysbios/hal/Hwi.h>

**#include** <ti/sysbios/knl/Task.h>

**#include** <ti/sysbios/knl/Event.h>

**#include** <ti/sysbios/knl/Semaphore.h>

**#include** <ti/drivers/Board.h>

Clock\_Handle clkHandler;

Clock\_Struct clkStruct;

//Hardware handler

Hwi\_Handle ADChwiHandler;

Hwi\_Struct ADChwistruct;

// create SWI handler

Swi\_Handle swiHandle;

Swi\_Struct swistruct;

//set pt value : Event 1

Event\_Struct Set\_pt\_value\_Struct;

Event\_Handle Set\_pt\_value\_Struct\_Handle;

//new setpoint

Event\_Struct new\_setpoint\_value\_Struct;

Event\_Handle new\_setpoint\_Struct\_Handle;

//tasks

//Run\_inpiut task

Task\_Struct inputTaskStruct;

Char inputTaskStack[TASKSTACKSIZE];

//update lcd

Task\_Struct updateLCDTaskStruct;

Char updateLCDTaskStack[TASKSTACKSIZE];

//temp

**int** temperature = 0;

**#endif**

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

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;

lcd\_init();

adc\_init();

// TI-RTOS

/\* Construct BIOS Objects \*/

/\* Tasks

\* Events

\* Semaphore

\* Clocks

\* SWI

\* HWI

\*/

//Clock

Clock\_Params clkParams;

//hardware interrupt objects

Hwi\_Params ADChwiParams;

Error\_Block ADCeb;

//Enable the HWi

//From B.3 figure globally enables the hardware interrupts.

Hwi\_enable();

//Software interrupt objects

Swi\_Params ADCswiParams;

//task parsms

Task\_Params taskParams;

/\* Call driver init functions \*/

Board\_init();

//init the cloock

Clock\_Params\_init(&clkParams);

//set the start flag of the flag true

clkParams.startFlag = TRUE;

//constructing the clock

Clock\_construct(&clkStruct,

(Clock\_FuncPtr)start\_conversion,

CLOCK\_PERIOD \* 2,

&clkParams);

//handeling of the clock

clkHandler = Clock\_handle(&clkStruct);

//set the period of the clock

Clock\_setPeriod(clkHandler, CLOCK\_PERIOD);

//Init the HWI for A/D IRQ Handler

//Based on the technical reference manuel

/\*

\* Creating HWI object

\* HWI\_Handle hwi0;

\* Hwi\_Parms hwiParams;

\* Error\_Block eb;

\*

\* init the error code

\* run Hwi\_Params\_init

\*

\* set the parsms arguement value

\* create Hwi

\*/

Error\_init(&ADCeb);

Hwi\_Params\_init(&ADChwiParams);

ADChwiParams.priority = 1; //Top priority

ADChwiHandler = Hwi\_create

(

RTOS\_ADC\_INTERRUPT\_NUMBER,

(Hwi\_FuncPtr)ADC\_HWI\_FUNC,

&ADChwiParams,

&ADCeb

);

/\*

\* Setup of the SWI for A/D Converter

\*

\*/

Swi\_Params\_init( &ADCswiParams );

ADCswiParams.priority = 2; //second top priority

ADCswiParams.trigger = 0; //trigger value

ADCswiParams.arg0 = 1;

ADCswiParams.arg1 = 0;

Swi\_construct

(

&swistruct,

(Swi\_FuncPtr)ADC\_SWI\_FUNC,

&ADCswiParams,

NULL

);

swiHandle = Swi\_handle(&swistruct);

//setup for event when new pt value is possible

Event\_construct(&Set\_pt\_value\_Struct, NULL);

Event\_construct(&new\_setpoint\_value\_Struct, NULL);

/\* Obtain event instance handle \*/

Set\_pt\_value\_Struct\_Handle = Event\_handle(&Set\_pt\_value\_Struct);

new\_setpoint\_Struct\_Handle = Event\_handle(&new\_setpoint\_value\_Struct);

/\* Construct writer/reader Task threads \*/

// Setup the run inputs task

// higher priority than LCD update task

Task\_Params\_init(&taskParams);

taskParams.stackSize = TASKSTACKSIZE;

taskParams.priority = 3;

taskParams.stack = &inputTaskStack;

Task\_construct(&inputTaskStruct, (Task\_FuncPtr)Process\_Inputs,

&taskParams, NULL);

//setup for update LCD display task

taskParams.stackSize = TASKSTACKSIZE;

taskParams.stack = &updateLCDTaskStack;

taskParams.priority = 4;

Task\_construct(&updateLCDTaskStruct, (Task\_FuncPtr)Update\_LCD,

&taskParams, NULL);

/\* start the BIOS \*/

BIOS\_start();

**return**(0);

} // end main

/\*

\* Hardware Interrupt function

\* called when a new ADC value is ready.

\* signals to a SWI when a conversion is ready

\*/

**void** **ADC\_HWI\_FUNC**( **void** )

{

**int** readIV = ADC14->IV; // Reading the IV register to clear

uint32\_t adc\_val = ADC14->MEM[0];

Swi\_post(swiHandle);

//printf("Inside HWI function\n");

}

/\*

\* Software interrupt function

\* Reads the ADC value, converts it to the temperature range 50-90F

\*/

**void** **ADC\_SWI\_FUNC**( UArg arg0, UArg arg1 )

{

//calculate the value from ADC14 and converts to temp

temperature = ((**float**)ADC14->MEM[0])/ADC\_12\_BIT\_RES;

//temperature before this equation will be around 40 ticks

temperature = ( temperature ) + 50;

//printf("Inside SWI function\n");

//printf("Running the event\n");

Event\_post( Set\_pt\_value\_Struct\_Handle, Event\_Id\_00 );

}

//process the inputs based on the event id value

**void** **Process\_Inputs**( UArg arg0, UArg arg1 )

{

uint8\_t posted;

**static** **int** previous\_temp = 0;

**while**(1)

{

posted = Event\_pend( Set\_pt\_value\_Struct\_Handle,

Event\_Id\_00,

Event\_Id\_NONE,

BIOS\_WAIT\_FOREVER

);

//check for the posted value

**if**( posted == Event\_Id\_00)

{

**if**( temperature != previous\_temp)

{

//update previous temp

previous\_temp = temperature;

Event\_post( new\_setpoint\_Struct\_Handle, Event\_Id\_01 );

}

}

}

}

**void** **Update\_LCD**( UArg arg0, UArg arg1 )

{

uint8\_t lcd\_post;

**while**(1)

{

lcd\_post = Event\_pend( new\_setpoint\_Struct\_Handle,

Event\_Id\_01,

Event\_Id\_NONE,

BIOS\_WAIT\_FOREVER

);

//check for the posted value if true then update lcd

**if**( lcd\_post == Event\_Id\_01)

{

LCD\_goto\_xy(5 , 2);

LCD\_print\_udec3(temperature);

}

}

}

## EE4930\_LAB6.c

/\*

\* EE4930\_LAB6.c

\*

\* Created on: Feb 2, 2023

\* Author: jurado-garciaj

\*

\*

\* Description:

\* Reads from a potentiometer with an ADC. ADC reading is

\* periodically taken using a TI RTOS clock

\*

\*\*\*\*\*\*\*\*\* Nokia LCD interface reference \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Red SparkFun Nokia 5110 (LCD-10168)

\* -----------------------------------

\* Signal (Nokia 5110) LaunchPad pin

\* 3.3V (VCC, pin 1) power

\* Ground (GND, pin 2) ground

\* UCA3STE (SCE, pin 3) connected to P9.4

\* Reset (RST, pin 4) connected to P9.3

\* Data/Command (D/C, pin 5) connected to P9.2

\* UCA3SIMO (DN, pin 6) connected to P9.7

\* UCA3CLK (SCLK, pin 7) connected to P9.5

\* back light (LED, pin 8) not connected

\*

\* POTENTIOMETER reference

\* GND (CCW, pin 1) ground

\* Wiper (Wiper, pin 2) connected to

\* VCC (CW, pin 3) power

\*/

**#include** "EE4930\_LAB6.h"

**#include** "msp.h"

**#include** <stdio.h>

**#include** <string.h>

**#include** "msoe\_lib\_clk.h"

**#include** "msoe\_lib\_lcd.h"

**#include** "msoe\_lib\_delay.h"

**#include** <MSP432P401R\_GPIO.h>

**void** **lcd\_init**( **void** )

{

LCD\_Config();

LCD\_clear();

LCD\_home();

LCD\_contrast(10);

LCD\_goto\_xy(0 , 2);

LCD\_print\_str("Temp: xx F");

}

**void** **adc\_init**( **void** )

{

//A0

GPIO\_setAsInputPin(GPIO\_PORT\_P5, GPIO\_PIN5); //Set P5.5 as input for A0

// input on A0 into MEM0

P5->SEL0 |= BIT5; // use with A0

P5->SEL1 |= BIT5; // use with A0

// Setting up the control register

ADC14->CTL0 |= ADC14\_CTL0\_SHT0\_5 | ADC14\_CTL0\_SHP |

ADC14\_CTL0\_SSEL\_\_SMCLK | ADC14\_CTL0\_ON |

ADC14\_CTL0\_CONSEQ\_0;

// Sampling time, S&H=96, ADC14 on, SMCLK, with 10 Bit Resolutions

// make sure read the reset operation first

// CSTARTADDx this gets the start address, these bits select which

//ADC14 conversion

// memory registers is used for a single conversion or for the

//first conversion in a sequence.

// setting the memory to MEM0

ADC14->CTL1 &= ~ADC14\_CTL1\_RES\_3;

ADC14->CTL1 |= ADC14\_CTL1\_RES\_2;

ADC14->CTL1 |= (0 << ADC14\_CTL1\_CSTARTADD\_OFS); //A0

//Memory conversion control 0 register being set to get data for A0

ADC14->MCTL[0] |= ADC14\_MCTLN\_INCH\_0;

// Enabled interrupt information for IER0 register

ADC14->IER0 |= ADC14\_IER0\_IE0;

// enable the conversion

ADC14->CTL0 |= ADC14\_CTL0\_ENC;

}

**void** **start\_conversion**( **void** )

{

ADC14->CTL0 |= 1; // start ADC conversions

}

## EE4930\_LAB6.h

/\*

\* EE4930\_LAB6.h

\*

\* Created on: Feb 2, 2023

\* Author: jurado-garciaj

\*

\* Description:

\* Reads from a potentiometer with an ADC. ADC reading is

\* periodically taken using a TI RTOS clock

\*

\*\*\*\*\*\*\*\*\* Nokia LCD interface reference \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Red SparkFun Nokia 5110 (LCD-10168)

\* -----------------------------------

\* Signal (Nokia 5110) LaunchPad pin

\* 3.3V (VCC, pin 1) power

\* Ground (GND, pin 2) ground

\* UCA3STE (SCE, pin 3) connected to P9.4

\* Reset (RST, pin 4) connected to P9.3

\* Data/Command (D/C, pin 5) connected to P9.2

\* UCA3SIMO (DN, pin 6) connected to P9.7

\* UCA3CLK (SCLK, pin 7) connected to P9.5

\* back light (LED, pin 8) not connected

\*

\* POTENTIOMETER reference

\* GND (CCW, pin 1) ground

\* Wiper (Wiper, pin 2) connected to

\* VCC (CW, pin 3) power

\*

\*/

**#ifndef** APP\_EE4930\_LAB6\_H\_

**#define** APP\_EE4930\_LAB6\_H\_

**#include** <xdc/std.h>

**#define** CLOCK\_PERIOD 10

**#define** RTOS\_ADC\_INTERRUPT\_NUMBER 40

**#define** ADC\_12\_BIT\_RES (102)

**#define** TASKSTACKSIZE 512

//Init the lcd

**void** **lcd\_init**( **void** );

**void** **adc\_init**( **void** );

**void** **start\_conversion**( **void** );

**void** **ADC\_HWI\_FUNC**( **void** );

**void** **ADC\_SWI\_FUNC**( UArg arg0, UArg arg1 );

**void** **Process\_Inputs**( UArg arg0, UArg arg1 );

**void** **Update\_LCD**( UArg arg0, UArg arg1 );

**#endif** /\* APP\_EE4930\_LAB6\_H\_ \*/