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 4 : “Finite State Machine”

Name: Jorge Jurado-Garcia

Courses: EE 4930 011

Date: 01/20/2023

# Objectives

This lab objective is to implement a Finite state machine using the look up table with function pointers mechanism.

# Description

For this lab, I implemented three structs and enumeration for my project. The first struct has four main data types which are eSystemState, eSystemEvent, and the Event Handler. This structure is what will be used for my project and get referenced throughout the whole project. The next structure is where the System Inputs and System Outputs which were declared in the Laboratory 4 documentation. Example of some inputs and outputs are Humidity Setpoints, Fan Control, Compressor, and Ice Sensor. Finally, the last structure was the System itself which included many of the needed data types to make the code work. An example below shows the declaration of the Dehumidifier structure.

//structure of state and event with event handler

**typedef** **struct**

{

eSystemState eStateMachine;

eSystemEvent eStateMachineEvent;

pfEventHandler pfStateMachineEvnentHandler;

} sStateMachine;

# Conclusion

Overall, this lab was a success, I learned a lot about developing a loop up table and writing the code I found some of the main reasoning in which State Machines with a loop up table. Which was the easiness of changing the different states in your code and code understanding. Using this method instead of the simple case statements and if nested statement makes the code for complex but easier to modify and read once you have a solid understanding of pointers, arrays, and function calls by reference. I really enjoyed this lab and creating my polling function which was the hardest part of the lab for me.

# Attachments

## Schematic

Diagram, schematic

Description automatically generated

## State Machine Flow Chart

Diagram

Description automatically generated

## Main

/\* Copyright (C) 2022 MSOE

\*

\* All Rights Reserved

\* You may not use, distribute or modify this code without the

\* express written permission of MSOE

\*

\* Contact Info

\* jorgejuradogarcia2@gmail.com

\* 608-312-5950

\*

\*/

**#include** <stdio.h>

**#include** "msp432.h"

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

**#include** "defines.h"

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

//Preprocessors

**#define** LAB4\_ON TRUE

**#define** CLEAR 21

// Lab modules

**#if** LAB4\_ON

**#include** "EE4930\_LAB4.h"

**#endif**

/\*

\* main.c

\*/

//global variables for Lab 2

**unsigned** adc\_val;

**float** ADC\_Percentage;

**float** PWM\_dutycycle;

**int** Calculated\_dutycycle;

//global variables for Lab4

**unsigned** adc\_val\_A0;

**unsigned** adc\_val\_A2;

**float** ADC\_A2\_Percentage;

**float** ADC\_A0\_Percentage;

**unsigned** **char** ticks;

eSystemInputs Inputs;

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

**#if** LAB4\_ON

Clock\_Init\_48MHz();

//Interrupt setup

NVIC->ISER[0] |= ( 1<<24 ); //NVIC for ADC14 at ISER[24]

NVIC->ISER[1] |= ( 1 << 3 ); //NVIC for PORT 1 at ISER[35]

NVIC->ISER[0] |= ( 1<<25 ); //NVIC for TIMER\_32\_1 at ISER[25]

\_\_enable\_interrupt();

Dehumidifier\_init();

eSystemEvent eNewEvent = *Last\_event*;

eSystemState eNextState = *Off\_state*;

**while**(1)

{

**if**( ticks == TICKS\_MAX)

{

//read the inputs

Dehumidifier\_Read(Inputs);

//Run the state machine

eNewEvent = Dehumidifier\_ReadEvent();

//poll of the state machine

eNextState = Dehumidifier\_Poll(eNewEvent, eNextState );

}

}

**#endif**

} // end main

**#if** LAB4\_ON

// Interrupt Handler for ADC14

// Will enter into this interrupt source with the Interrupt of the ADC14 was set

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

{

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

// read A/D result

**switch**( readIV )

{

**case** 0x0C:

adc\_val\_A0 = ADC14->MEM[0];

ADC\_A0\_Percentage = 100 \* (**float**)adc\_val\_A0/16384;

//reading humidiity

Inputs.Humidity\_percentage = ADC\_A0\_Percentage;

**break**;

**case** 0x10:

adc\_val\_A2 = ADC14->MEM[2];

ADC\_A2\_Percentage = (**float**)adc\_val\_A2/16384;

//READING Room Temperature

// 40-90 degrees so the values can range from 50 points

Inputs.Room\_temperature = (**int**)(ADC\_A2\_Percentage\*50) + 40;

**break**;

}

}

// Interrupt Handler for SW1

**void** **PORT1\_IRQHandler**()

{

**int** readIV = P1->IV; // Reading IV register to clear

**switch**( readIV )

{

**case** 0x04: // P1.1

Inputs.Humidity\_setpoints += 5;

**break**;

**case** 0x0A: // P1.4

Inputs.Humidity\_setpoints -= 5;

**break**;

}

**if**(Inputs.Humidity\_setpoints > 100)

{

Inputs.Humidity\_setpoints = 100;

}

}

//Interrupt Handler for TIMRE32\_Int1

**void** **T32\_INT1\_IRQHandler**()

{

TIMER32\_1->INTCLR = CLEAR;

// Read the Ice sensor pin

Inputs.Ice\_sensor = ( (P5->IN & GPIO\_PIN4) >> 4 );

// run adc converter

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

ticks++;

**if**( ticks > TICKS\_MAX)

{

ticks = 0;

}

}

**#endif**

## EE4930\_LAB4.c

/\*

\* EE4930\_LAB4.c

\*

\* Created on: Jan 2, 2023

\* Author: jurado-garciaj

\*/

// Included files

**#include** <stdio.h>

**#include** "msp432.h"

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

**#include** "defines.h"

**#include** "EE4930\_LAB4.h"

// Drivers

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

sDehumidifier Honey;

//private function prototypes

**static** **void** **Initlab4\_A2D**( **void** );

**static** **void** **setup\_outputs**( **void** );

**static** **void** **setup\_inputs**( **void** );

//function call to dispatch the amount and return the ideal state

**static** eSystemState **Humidity\_above\_setpoint\_Ice\_off\_Handler**(**void**);

**static** eSystemState **Humidity\_below\_setpoint\_Ice\_off\_Handler**(**void**);

**static** eSystemState **Humidity\_within\_setpoint\_Ice\_off\_Handler**(**void**);

**static** eSystemState **Ice\_Sensed\_Handler**(**void**);

**static** eSystemState **Ice\_Not\_Senses\_Handler**(**void**);

//public function prototypes

**void** **Dehumidifier\_Read**( eSystemInputs inputs );

eSystemEvent **Dehumidifier\_ReadEvent**( **void** );

eSystemState **Dehumidifier\_Poll**( eSystemEvent NewEvent, eSystemState NextState);

**void** **Dehumidifier\_init**( **void** );

//module specific structure

//Initialize array of structure with states and event with proper handler

sStateMachine sMachiney [LOOPUP\_ROW][LOOPUP\_COLUMN] =

{

{

//off state moves

{*Off\_state*,*Ice\_sensed\_event*,Ice\_Sensed\_Handler},

{*Off\_state*,*Humidity\_above\_setpoint\_Ice\_off\_event*,Humidity\_above\_setpoint\_Ice\_off\_Handler},

{*Off\_state*,*Humidity\_within\_setpoint\_Ice\_off\_event*,Humidity\_within\_setpoint\_Ice\_off\_Handler},

{*Off\_state*, *Humidity\_below\_setpoint\_Ice\_off\_event*, Humidity\_below\_setpoint\_Ice\_off\_Handler}

},

{

//Defrost state moves

{*Defrost\_state*,*Ice\_off\_event*,Ice\_Not\_Senses\_Handler},

{*Defrost\_state*,*Ice\_sensed\_event*,Ice\_Sensed\_Handler},

{*Defrost\_state*,*Humidity\_below\_setpoint\_Ice\_off\_event*,Humidity\_below\_setpoint\_Ice\_off\_Handler},

{*Defrost\_state*, *Humidity\_above\_setpoint\_Ice\_off\_event*, Humidity\_above\_setpoint\_Ice\_off\_Handler}

},

{

//active state moves

{*Active\_state*,*Ice\_sensed\_event*,Ice\_Sensed\_Handler},

{*Active\_state*,*Humidity\_below\_setpoint\_Ice\_off\_event*,Humidity\_below\_setpoint\_Ice\_off\_Handler},

{*Active\_state*,*Humidity\_within\_setpoint\_Ice\_off\_event*,Humidity\_within\_setpoint\_Ice\_off\_Handler},

{*Active\_state*, *Humidity\_above\_setpoint\_Ice\_off\_event*, Humidity\_above\_setpoint\_Ice\_off\_Handler}

},

};

//state machine info

//function call to dispatch the amount and return the ideal state

eSystemState **Humidity\_above\_setpoint\_Ice\_off\_Handler**(**void**)

{

//run outputs

//turn on fan

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P4, GPIO\_PIN1 ); //fan

//Turn the compressor on

// make it a green led

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P2, GPIO\_PIN1 ); //Green

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN0 ); //Red

//turn the defrost mode off

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN2 ); //Green

LCD\_goto\_xy(0,1);

LCD\_print\_str("FAN/COMP ");

LCD\_goto\_xy(0,2);

LCD\_print\_str("ON/ON ");

Honey.sOutputs.Fan\_control = ON;

Honey.sOutputs.Compressor = ON;

Honey.sOutputs.LCD\_Display\_flag = TRUE;

//Honey.eCurrentState = Active\_state;

**return** *Active\_state*;

}

eSystemState **Humidity\_below\_setpoint\_Ice\_off\_Handler**(**void**)

{

//run outputs

//turn off fan

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P4, GPIO\_PIN1 ); //fan

//Turn the compressor off

// make it a red led

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN1 ); //Green

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P2, GPIO\_PIN0 ); //Red

//turn the defrost mode off

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN2 ); //Green

//show on LCD we are in defrost mode

LCD\_goto\_xy(0,1);

LCD\_print\_str("FAN/COMP ");

LCD\_goto\_xy(0,2);

LCD\_print\_str("OFF/OFF ");

Honey.sOutputs.Fan\_control = OFF;

Honey.sOutputs.Compressor = OFF;

Honey.sOutputs.LCD\_Display\_flag = TRUE;

//Honey.eCurrentState = Off\_state;

**return** *Off\_state*;

}

eSystemState **Humidity\_within\_setpoint\_Ice\_off\_Handler**(**void**)

{

//run outputs and check if this true

// keep the same outputs as from before

// just change the LCD update to true still

//turn off the defrost mode led

GPIO\_setOutputLowOnPin(GPIO\_PORT\_P2, GPIO\_PIN2 ); //Blue

Honey.sOutputs.LCD\_Display\_flag = TRUE;

Honey.eCurrentState = Honey.eCurrentState;

**return** Honey.eCurrentState;

}

eSystemState **Ice\_Sensed\_Handler**(**void**)

{

//run outputs and check if this true

//if Ice sensed turn on the fan

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P4, GPIO\_PIN1 );

//Turn the compressor off

// make it a red led

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN1 ); //Green

GPIO\_setOutputHighOnPin( GPIO\_PORT\_P2, GPIO\_PIN0 ); //Red

//show on LCD we are in defrost mode

GPIO\_setOutputHighOnPin(GPIO\_PORT\_P2, GPIO\_PIN2 ); //Blue

LCD\_goto\_xy(0,1);

LCD\_print\_str("FAN/COMP ");

LCD\_goto\_xy(0,2);

LCD\_print\_str("ON/OFF ");

Honey.sOutputs.Fan\_control = ON;

Honey.sOutputs.Compressor = OFF;

Honey.sOutputs.LCD\_Display\_flag = TRUE;

// Honey.eCurrentState = Defrost\_state;

**return** *Defrost\_state*;

}

eSystemState **Ice\_Not\_Senses\_Handler**(**void**)

{

// run outputs since we are not sensing ice no more and move to its next state

//show on LCD we not in defrost mode

//BLUE

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN2);

Honey.sOutputs.LCD\_Display\_flag = TRUE;

//need to figure out how to fill in this previous state value

// this can be done by configuring the inputs

**if**( Honey.sInputs.Humidity\_percentage > (Honey.sInputs.Humidity\_setpoints + 5) )

{

//humidity percentage is greater than set points return to dehumid.

**return** *Active\_state*;

}

**return** Honey.ePreviousState;

}

eSystemState **Dehumidifier\_Poll**( eSystemEvent NewEvent, eSystemState NextState)

{

**static** eSystemState State = *last\_state*;

//Check NULL pointer and array boundary

**if**( ( NextState < *last\_state*) && (NewEvent < *Last\_event*) )

{

**int** i = 0;

//loop around the state machine until the state was found

**for**(i=0; i < LOOPUP\_COLUMN; i ++ )

{

//if the Event machines and Stata machine Event Handel also matches

**if**( (sMachiney[NextState][i].eStateMachineEvent == NewEvent ) &&

(sMachiney[NextState][i].pfStateMachineEvnentHandler != NULL ) )

{

State = NextState;

// function call as per the state and event and return the next state of FSM

NextState = (\*sMachiney[NextState][i].pfStateMachineEvnentHandler)();

Honey.Errorcode = FALSE;

//only update previous state if we are not in defrost mode. We will need this state back

**if**( NextState != *Defrost\_state*)

{

//save the previous state;

Honey.ePreviousState = State;

}

Honey.eCurrentState = NextState;

}

}

}

**else**

{

//Invalid

Honey.Errorcode = TRUE;

}

**return** NextState;

}

//Read the event

eSystemEvent **Dehumidifier\_ReadEvent**( **void** )

{

//check if we are in the defrost state

**if**( Honey.eCurrentState == *Defrost\_state* )

{

//if sensor is false go to the previous state in which it was in

**if**( Honey.sInputs.Ice\_sensor == FALSE)

{

**return** *Ice\_off\_event*;

}

**return** *Ice\_sensed\_event*;

}

//if we are not in the defrost current state mode

//Based on the inputs check which even to trigger

**if**( Honey.sInputs.Ice\_sensor == TRUE)

{

**return** *Ice\_sensed\_event*;

}

**else**

{

//check if the Sensor readings

**if**( Honey.sInputs.Humidity\_percentage > (Honey.sInputs.Humidity\_setpoints + 5))

{

//humidity percentage is greater than set points

**return** *Humidity\_above\_setpoint\_Ice\_off\_event*;

}

**if**( Honey.sInputs.Humidity\_percentage < (Honey.sInputs.Humidity\_setpoints - 5))

{

//humidity percentage is greater than set points

**return** *Humidity\_below\_setpoint\_Ice\_off\_event*;

}

//if neither of these statements are true then we are in idle mode

**return** *Humidity\_within\_setpoint\_Ice\_off\_event*;

}

}

//Read the global variable from

**void** **Dehumidifier\_Read**( eSystemInputs inputs )

{

Honey.sInputs.Humidity\_percentage = inputs.Humidity\_percentage;

Honey.sInputs.Humidity\_setpoints = inputs.Humidity\_setpoints;

Honey.sInputs.Ice\_sensor = inputs.Ice\_sensor;

Honey.sInputs.Room\_temperature = inputs.Room\_temperature;

//update LCD Humidity percentages, set-points, and room temperature

LCD\_goto\_xy(8,3);

LCD\_print\_udec3(Honey.sInputs.Room\_temperature);

LCD\_goto\_xy(8,4);

LCD\_print\_udec3(Honey.sInputs.Humidity\_setpoints);

LCD\_goto\_xy(8,5);

LCD\_print\_udec3(Honey.sInputs.Humidity\_percentage);

}

// Init the dehumidifer

**void** **Dehumidifier\_init**( **void** )

{

Honey.eCurrentState = *Off\_state*;

Honey.ePreviousState = *Off\_state*;

Honey.sInputs.Humidity\_percentage = 0;

Honey.sInputs.Humidity\_setpoints = 0;

Honey.sInputs.Ice\_sensor = FALSE;

Honey.sInputs.Room\_temperature = 0;

Honey.sOutputs.Compressor = OFF;

Honey.sOutputs.Fan\_control = OFF;

Honey.sOutputs.LCD\_Display\_flag = OFF;

setup\_inputs();

setup\_outputs();

**return**;

}

//Init the setup\_outputs

**void** **setup\_outputs**( **void** )

{

//Fan control Output control turn on 5V DC fan 4.1

GPIO\_setAsOutputPin( GPIO\_PORT\_P4, GPIO\_PIN1 );

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P4, GPIO\_PIN1 );

P4->DS |= BIT1;

// Compressor output by using MSP on board led

//Red

GPIO\_setAsOutputPin( GPIO\_PORT\_P2, GPIO\_PIN0 );

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN0 );

//Green

GPIO\_setAsOutputPin( GPIO\_PORT\_P2, GPIO\_PIN1 );

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN1 );

//BLUE

GPIO\_setAsOutputPin( GPIO\_PORT\_P2, GPIO\_PIN2 );

GPIO\_setOutputLowOnPin( GPIO\_PORT\_P2, GPIO\_PIN2);

// Configures the LCD display screen clears display afterwards

LCD\_Config();

//Printing "EE4930-011" on the first line of the LCD screen

LCD\_goto\_xy(0,1);

LCD\_print\_str("FAN/COMP");

LCD\_goto\_xy(0,2);

LCD\_print\_str("OFF ");

LCD\_goto\_xy(0,3);

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

LCD\_goto\_xy(0,4);

LCD\_print\_str("Setpoint:");

LCD\_goto\_xy(0,5);

LCD\_print\_str("Humidity:");

**return**;

}

//Init the setup\_inits

**void** **setup\_inputs**( **void** )

{

Stop\_watchdog(); // stop Watch dog timer

Set\_ports\_to\_out(); //sets all the ports to outputs to prevent floating inputs

// For Humidity Setpoint - two pushbutton switches

// SW1

GPIO\_setAsInputPinWithPullUpResistor( GPIO\_PORT\_P1, GPIO\_PIN1 );

//Set P1.1 as an Input with Pull up resistor high

P1->IES &= ~GPIO\_PIN1; // enable edge select

P1->IE |= GPIO\_PIN1; // enable interrupt

// SW2

GPIO\_setAsInputPinWithPullUpResistor( GPIO\_PORT\_P1, GPIO\_PIN4 );

//Set P1.4 as an Input with Pull up resistor high

P1->IES &= ~GPIO\_PIN4; // enable edge select

P1->IE |= GPIO\_PIN4; // enable interrupt

// Potentiometer for Room Temperature

//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

// Potentiometer for Humidity

//A2

GPIO\_setAsInputPin(GPIO\_PORT\_P5, GPIO\_PIN3); //Set P5.3 as input for A2

// input on A2 into MEM2

P5->SEL0 |= BIT3; // use with A2

P5->SEL1 |= BIT3; // use with A2

// Ice Senior - yes.no (switch or jumper) //setup pin P5.4 as input

GPIO\_setAsInputPin(GPIO\_PORT\_P5, GPIO\_PIN4); //Set P5.4 as input ice stuff

Initlab4\_A2D();

//init Timer32 for ice sensor

// Set master clock (MCLK) to HFXTCLK with divide by 1 - 48MHz

TIMER32\_1->LOAD |= 46875; //Value in which timer will get reloaded

/\*

\* TIMER CLOCK SPEED: 48MHz MHz

\* TImer disabled

\* Periodic mode Enabled

\* Timer Interrupt enabled

\* Clock division of 256

\* Clock After Division in 48MHz/256 = 0.1875 MHz = 187,500 Hz

\* Size of 16-bit counter

\* Ticks 16 bits = 65535 ticks

\* Timer\_sec = 1/187500 = 5.333\*10^(-6)

\* Timer After expiration = 5.333\*10^(-6) \* 46875 = 0.25 seconds

\* Wrapping mode

\* 32-bit counter

\*/

/\*

\*

\*/

TIMER32\_1->CONTROL |= 0x6A;

// ENABLE TIMER

TIMER32\_1->CONTROL |= 0x80;

**return**;

}

//init for A2D converter

**void** **Initlab4\_A2D**( **void** )

{

// Setting up the control register

ADC14->CTL0 |= ADC14\_CTL0\_SHT0\_5 | ADC14\_CTL0\_SHP | ADC14\_CTL0\_SSEL\_\_SMCLK | ADC14\_CTL0\_ON | ADC14\_CTL0\_CONSEQ\_1;

// Sampling time, S&H=96, ADC14 on, SMCLK, with 14 Bit Resolutions

// make sure read the reset operation first

// CSTARTADDx this gets the start address.

// 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\_3;

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

//ADC14->CTL1 |= (2 << ADC14\_CTL1\_CSTARTADD\_OFS); //A2

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

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

//Memory conversion control 2 register being set to get data for A2

ADC14->MCTL[2] |= ( ADC14\_MCTLN\_INCH\_2 | ADC14\_MCTLN\_EOS) ;

// Enabled interrupt information for IER0 register

ADC14->IER0 |= (ADC14\_IER0\_IE0 | ADC14\_IER0\_IE2 );

// enable the conversion

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

**return**;

}

## EE4930\_LAB4.h

/\*

\* EE4930\_LAB4.h

\*

\* Created on: Jan 2, 2023

\* Author: jurado-garciaj

\* Run a timer to read the Freezer inputs to see when to two tick

\*/

**#ifndef** APP\_EE4930\_LAB4\_H\_

**#define** APP\_EE4930\_LAB4\_H\_

**#define** FALSE 0

**#define** TRUE 1

**#define** OFF 0

**#define** ON 1

**#define** TICKS\_MAX 9

**#define** LOOPUP\_COLUMN 4

**#define** LOOPUP\_ROW 3

//Different states

**typedef** **enum**

{

*Off\_state*,

*Defrost\_state*,

*Active\_state*,

*last\_state*

} eSystemState;

//Different type events

**typedef** **enum**

{

*Humidity\_above\_setpoint\_Ice\_off\_event*,

*Humidity\_below\_setpoint\_Ice\_off\_event*,

*Humidity\_within\_setpoint\_Ice\_off\_event*,

*Ice\_sensed\_event*,

*Ice\_off\_event*,

*Last\_event*

} eSystemEvent;

//input structure of state machine

**typedef** **struct**

{

**unsigned** **char** Room\_temperature;

**unsigned** **char** Humidity\_percentage;

**unsigned** **char** Humidity\_setpoints;

**char** Ice\_sensor;

} eSystemInputs;

//output structure of state machine

**typedef** **struct**

{

**char** Fan\_control;

**char** Compressor;

**char** LCD\_Display\_flag;

} eSystemOutputs;

//typedef of function pointer

**typedef** eSystemState (\*pfEventHandler)(**void**);

//structure of state and event with event handler

**typedef** **struct**

{

eSystemState eStateMachine;

eSystemEvent eStateMachineEvent;

pfEventHandler pfStateMachineEvnentHandler;

} sStateMachine;

//structure of module itself

**typedef** **struct**

{

eSystemInputs sInputs;

eSystemOutputs sOutputs;

eSystemState eCurrentState;

eSystemState ePreviousState;

**char** Errorcode;

} sDehumidifier;

**void** **Dehumidifier\_Read**( eSystemInputs inputs );

eSystemEvent **Dehumidifier\_ReadEvent**( **void** );

eSystemState **Dehumidifier\_Poll**( eSystemEvent NewEvent, eSystemState NextState);

**void** **Dehumidifier\_init**( **void** );

**#endif** /\* APP\_EE4930\_LAB4\_H\_ \*/