```
/*
 *
 *      Activate_LED_Control_ASM.asm
 *
 *
 */

.extern  __Z28FastForward_OneSimulationTICP16COFFEEPOT_DEVICE;

.section program;

.global _Activate_LED_Control_ASM;
_Activate_LED_Control_ASM:
 #define portPattern_R0 R0
 #define portPointer_P0 P0
 #define temp_R1 R1

 #define INPUT_MASK 0x0002
 #define LED_MASK 0x0fff


        LINK 16;
        P0 = R0;
        portPattern_R0 = W[portPointer_P0](Z);

        //pCoffeePot->controlRegister |= LED_POWER_ENABLE_BIT;
        temp_R1 = INPUT_MASK;
        R2 = portPattern_R0 | temp_R1;

        //pCoffeePot->controlRegister &= ~(0xf << LED_CONTROL_FIELD_OFFSET);
        temp_R1 = LED_MASK;
        R2 = R2 & temp_R1;

        [P0] = R2;

        CALL __Z28FastForward_OneSimulationTICP16COFFEEPOT_DEVICE;

        UNLINK;
_Activate_LED_Control_ASM.END:
        RTS;



/*
 *
 *      Demonstrate_LEDControl_ASM.asm
 *
 *
 */
 //.extern  __Z28FastForward_OneSimulationTICP16COFFEEPOT_DEVICE;

 .section program;

 .global _Demonstrate_LEDControl_ASM;

_Demonstrate_LEDControl_ASM:
        //LINK 16;
 #define inArg_R0 R0
 #define control_Register_Location R0
 #define CNTRL_VALUE R1
 #define LED_VALUE R2
 #define RET_VALUE R3
 #define LED_OFFSET 12
 #define LED_MASK 0x0fff

        //control_Register_Location = inArg_R0 + value_offset;
        P0 = R0;
        CNTRL_VALUE = W[P0](Z);
```

```
        LED_VALUE = CNTRL_VALUE >> LED_OFFSET;
        //in case leds arent what im expecting
        R3 = 1;
        CC = LED_VALUE == R3;
        if CC JUMP label_2;
label_1:
        RET_VALUE = 1;
        JUMP end_switch;
label_2:
        RET_VALUE = 2;
        //JUMP end_switch;

end_switch:
//now LED_VALUE (R2) is free
        //shift ret value to correct position in control reg
        RET_VALUE = RET_VALUE << LED_OFFSET;

        //set LEDs to 0
        R2 = LED_MASK;
        CNTRL_VALUE = CNTRL_VALUE & R2;

        RET_VALUE = RET_VALUE | CNTRL_VALUE;

        [P0] = RET_VALUE;

        //fast forward simulation
        //CALL __Z28FastForward_OneSimulationTICP16COFFEEPOT_DEVICE;

        //UNLINK;
_Demonstrate_LEDControl_ASM.END:
RTS;



/*********************************************************************
 * CoffeePot_assignment_1_Core0.cpp
 *********************************************************************/

#include <stdio.h>
#include <sys/platform.h>
#include <sys/adi_core.h>
#include <ccblkfn.h>
#include "adi_initialize.h"
#include "My_CoffeePot_Functions.h"

#include "../include/Sim_CoffeePot_Functions2018.h"
#include "../include/Sim_CoffeePot_SimulatorStructures2018.h"

//P 1, T 4, CR 0x1F, TW 255, HR 0, HRB 0, CT 89, mWL 306, CN 129
//P 2, T 4, CR 0x1F, TW 265, HR 0, HRB 0, CT 89, mWL 400, CN 129
//P 3, T 4, CR 0x1F, TW 315, HR 0, HRB 0, CT 90, mWL 478, CN 129
//P 4, T 4, CR 0x1F, TW 245, HR 0, HRB 0, CT 89, mWL 336, CN 129


#define MAX_SECONDS                             200

#define MAX_WATER_LEVEL                 306
#define MAX_WATER_LEVEL_POT_2   400
#define MAX_WATER_LEVEL_POT_3   478
#define MAX_WATER_LEVEL_POT_4   336

#define MAX_TEMPERATURE  92

//TESTING
//#define DESIGN_1 1
#define DESIGN_2 1
//#define DESIGN_3 1
#define FINAL_CONTROL 1
```

```
extern "C" void Demonstrate_LEDControl_ASM (COFFEEPOT_DEVICE*);
extern "C" void Activate_LED_Control_ASM (COFFEEPOT_DEVICE*);

char __argv_string[] = "";

int main(int argc, char *argv[])
{
        adi_initComponents();



        ShowNameProcessorUsed( );

        Init_CoffeePotSimulation(NUM_COFFEEPOTS);
        unsigned long int secondsCounter = 0;
        //TurnOffNotification(true);
#if NUM_COFFEEPOTS > 0
        COFFEEPOT_DEVICE *pMyCoffeePot1 = Add_CoffeePotToSystem_PlugAndPlay(COFFEEPOT1, "Keenan's
Coffeepot");

        Init_CoffeePot(pMyCoffeePot1);

        Activate_LED_Control_ASM(pMyCoffeePot1);
        Activate_Water_Control (pMyCoffeePot1);
        Activate_Heater_Control (pMyCoffeePot1);
#endif
#if NUM_COFFEEPOTS > 1
        COFFEEPOT_DEVICE *pMyCoffeePot2 = Add_CoffeePotToSystem_PlugAndPlay(COFFEEPOT2, "Sunny's
Coffeepot");

        Init_CoffeePot(pMyCoffeePot2);

        Activate_LED_Control(pMyCoffeePot2);
        Activate_Water_Control (pMyCoffeePot2);
        Activate_Heater_Control (pMyCoffeePot2);
#endif
#if NUM_COFFEEPOTS > 2
        COFFEEPOT_DEVICE *pMyCoffeePot3 = Add_CoffeePotToSystem_PlugAndPlay(COFFEEPOT3, "Rae's
Coffeepot");

        Init_CoffeePot(pMyCoffeePot3);

        Activate_LED_Control(pMyCoffeePot3);
        Activate_Water_Control (pMyCoffeePot3);
        Activate_Heater_Control (pMyCoffeePot3);
#endif
#if NUM_COFFEEPOTS > 3
        COFFEEPOT_DEVICE *pMyCoffeePot4 = Add_CoffeePotToSystem_PlugAndPlay(COFFEEPOT4, "Louis'
Coffeepot");

        Init_CoffeePot(pMyCoffeePot4);

        Activate_LED_Control(pMyCoffeePot4);
        Activate_Water_Control (pMyCoffeePot4);
        Activate_Heater_Control (pMyCoffeePot4);
#endif

#ifdef DESIGN_1

        bool continueDemonstrate_LEDControl = true;
        secondsCounter = 0;

        while (continueDemonstrate_LEDControl) {
                Demonstrate_LEDControl_CPP (pMyCoffeePot1);
                if (++secondsCounter >= MAX_SECONDS) continueDemonstrate_LEDControl = false;
                FastForward_OneSimulationTIC(pMyCoffeePot1);
                //ShowCoffeePotInformation(1, pMyCoffeePot1);
        }
```

```
#endif
#ifdef DESIGN_2

        bool continueDemonstrate_LEDControl_ASM = true;
        secondsCounter = 0;
        while (continueDemonstrate_LEDControl_ASM) {
                        Demonstrate_LEDControl_ASM (pMyCoffeePot1);
                        FastForward_OneSimulationTIC(pMyCoffeePot1);
                        if (++secondsCounter >= MAX_SECONDS /10) continueDemonstrate_LEDControl_ASM =
false;
        }


#endif
#ifdef DESIGN_3


        //TODO reactivate this
        bool continueDemonstrate_WaterControl = true;
        secondsCounter = 0;

        while (continueDemonstrate_WaterControl) {
                fillCoffeePotToWaterLevel (pMyCoffeePot1,MAX_WATER_LEVEL);
                if (++secondsCounter >= MAX_SECONDS) continueDemonstrate_WaterControl = false;
                //ShowCoffeePotInformation(1, pMyCoffeePot1);
        }


        bool continueDemonstrate_HeaterControl = true;
        secondsCounter = 0;

        while (continueDemonstrate_HeaterControl) {
                        HeatWaterToTemperature (pMyCoffeePot1,MAX_TEMPERATURE);
                        if (++secondsCounter >= MAX_SECONDS) continueDemonstrate_HeaterControl = false;
                        //ShowCoffeePotInformation(1, pMyCoffeePot1);
        }

#endif
#ifdef FINAL_CONTROL

        bool continueDemonstrate_Both_Temp_Water = true;
        secondsCounter = 0;

        while (continueDemonstrate_Both_Temp_Water) {
                        if (secondsCounter == 15 || secondsCounter == 120) {
                                if (secondsCounter == 15) printf("\n\n****Turning off all print
statements****\n\n");
                                else printf("\n\n****resuming all print statements****\n\n");
                                printCoffeePotInfo = !printCoffeePotInfo;
                                TurnOffNotification(!printCoffeePotInfo);
                        }
                Control_Both_Temp_Water (pMyCoffeePot1, MAX_WATER_LEVEL, MAX_TEMPERATURE);
#if NUM_COFFEEPOTS > 1
                Control_Both_Temp_Water (pMyCoffeePot2, MAX_WATER_LEVEL_POT_2, MAX_TEMPERATURE);
#endif
#if NUM_COFFEEPOTS > 2
                Control_Both_Temp_Water (pMyCoffeePot3, MAX_WATER_LEVEL_POT_3, MAX_TEMPERATURE);
#endif
#if NUM_COFFEEPOTS > 3
                Control_Both_Temp_Water (pMyCoffeePot4, MAX_WATER_LEVEL_POT_4, MAX_TEMPERATURE);
#endif
                FastForward_OneSimulationTIC(pMyCoffeePot1);
                if (++secondsCounter >= MAX_SECONDS) continueDemonstrate_Both_Temp_Water = false;
        }

#endif

        printf("Completed all tests.\n");
        return 0;
```

```
}


/*
 * My_CoffeePot_Functions.cpp
 *
 *  Created on: Oct 26, 2018
 *      Author: keenan.gaudio
 */

#include <stdio.h>
#include "../include/Sim_CoffeePot_Functions2018.h"
#include "../include/Sim_CoffeePot_SimulatorStructures2018.h"
#include "My_CoffeePot_Functions.h"

#define SHOW_FUNCTION_STUB_INFORMATION 0

extern "C" void Demonstrate_LEDControl_ASM (COFFEEPOT_DEVICE*);

void ShowNameProcessorUsed( void ){
#if defined(__ADSPBF533__)
        char processor[] = "__ADSPBF533__";
#else
        char processor[] = "__ADSPBF609__";
#endif
        printf ("CoffeePot running on %s system\n\n", processor);
}

inline void ShowFunctionStubInformation (char* functionNameInformation) {
        if (printCoffeePotInfo && SHOW_FUNCTION_STUB_INFORMATION) printf("%s\n", functionNameInformation);
}


void Init_CoffeePot( COFFEEPOT_DEVICE* pCoffeePot ){
        ShowFunctionStubInformation("Init_CoffeePot");
        OpenChannel_CoffeePot(pCoffeePot);
#if 0
        WriteControlRegister_CPP(pCoffeePot, 0x1);
#else
        // set the POWER_ON and INIT BIT in coffeepot control reg
        //unsigned short int oldCrtl = pCoffeePot->controlRegister;
        pCoffeePot->controlRegister |= INIT_STAY_POWERED_ON_BIT;
#endif
        CloseChannel_CauseSimulationTIC(pCoffeePot);

        while (!(pCoffeePot->controlRegister & DEVICE_READY_BIT_READ_ONLY))
FastForward_OneSimulationTIC(pCoffeePot);
        // read control reg in loop until bit 4 (init bit) is ready
        // exit when ready
        //if (SHOW_FUNCTION_STUB_INFORMATION) printf("leaving init with control reg : 0x%04x",pCoffeePot-
>controlRegister);

}

void Activate_LED_Control( COFFEEPOT_DEVICE* pCoffeePot ){
        ShowFunctionStubInformation("Activate_LED_Control");

        // turn on LED_POWER bit in ctrl reg without changing others
        //unsigned short int oldCrtl = pCoffeePot->controlRegister;
        pCoffeePot->controlRegister |= LED_POWER_ENABLE_BIT;
        //set LEDs to zero
        pCoffeePot->controlRegister &= ~(0xf << LED_CONTROL_FIELD_OFFSET);
        CloseChannel_CauseSimulationTIC(pCoffeePot);
         FastForward_OneSimulationTIC(pCoffeePot);
}

void Demonstrate_LEDControl_CPP ( COFFEEPOT_DEVICE* pCoffeePot ){
        ShowFunctionStubInformation("Demonstrate_LEDControl_CPP");
```

```
        //static unsigned short int state = 0;
        unsigned short int state = pCoffeePot->controlRegister >> LED_CONTROL_FIELD_OFFSET;
        //  Use state machine from Lab1
        //  state 0 -- turn on LED1 and off LED2
        //  state 1 -- turn on LED2 and off LED1
        switch(state) {
        case 0:
        case 2:
                        pCoffeePot->controlRegister |= LED1_CONTROL_BIT ;//on
                        pCoffeePot->controlRegister &= ~LED2_CONTROL_BIT ;       //off
                        break;
        case 1:
                        pCoffeePot->controlRegister &= ~LED1_CONTROL_BIT ;       //off
                        pCoffeePot->controlRegister |= LED2_CONTROL_BIT ;//on
                        break;
        default:
                pCoffeePot->controlRegister &= ~(0xf << LED_CONTROL_FIELD_OFFSET);
        }
         //FastForward_OneSimulationTIC(pCoffeePot);
}


void Activate_Water_Control ( COFFEEPOT_DEVICE* pCoffeePot ){
        ShowFunctionStubInformation("Activate_Water_Control");

        // turn on WATER POWER bit in ctrl reg without changing others
        //unsigned short int oldCrtl = pCoffeePot->controlRegister;
        pCoffeePot->controlRegister |= WATER_POWER_ENABLE_BIT;

        CloseChannel_CauseSimulationTIC(pCoffeePot);
         FastForward_OneSimulationTIC(pCoffeePot);
}

void fillCoffeePotToWaterLevel (COFFEEPOT_DEVICE* pCoffeePot, int wLevel){
        //TODO fix this
        ShowFunctionStubInformation("fillCoffeePotToWaterLevel");
        //  Activate water, fill to level (blocking)
                int currentWaterLevel = CurrentWaterLevel_CPP(pCoffeePot);
                int neededWater =  wLevel - currentWaterLevel;

                //if (CurrentTemperature_CPP(pCoffeePot) >= 79) neededWater+=10;

                if (neededWater > 255) neededWater = 255;
                else if (neededWater < 0) neededWater = 0;

                pCoffeePot->waterInFlowRegister  = neededWater;
//TODO while not if
        if (CurrentWaterLevel_CPP(pCoffeePot) < wLevel) {
                if(CurrentWaterLevel_CPP(pCoffeePot) > wLevel*4/5)
                        pCoffeePot->waterInFlowRegister = 1;
                FastForward_OneSimulationTIC(pCoffeePot);
        }

        pCoffeePot->waterInFlowRegister = 0;
}

void Activate_Heater_Control ( COFFEEPOT_DEVICE* pCoffeePot ){
        ShowFunctionStubInformation("Activate_Heater_Control");

        // turn on HEATER POWER bit in ctrl reg without changing others
        //unsigned short int oldCrtl = pCoffeePot->controlRegister;
        pCoffeePot->controlRegister |= HEATER_POWER_ENABLE_BIT;

        CloseChannel_CauseSimulationTIC(pCoffeePot);
         FastForward_OneSimulationTIC(pCoffeePot);
}

void HeatWaterToTemperature ( COFFEEPOT_DEVICE* pCoffeePot, int hLevel){
        //TODO fix this
```

```
        ShowFunctionStubInformation("HeatWaterToTemperature");
        // heat to temp (blocking)
        pCoffeePot->heaterBoostRegister = 1;
        pCoffeePot->heaterRegister = 165;
        while (CurrentTemperature_CPP(pCoffeePot) <( hLevel )) { //blocking version would have this as a
while loop
                pCoffeePot->heaterRegister = 255;
                if(CurrentTemperature_CPP(pCoffeePot) > hLevel*99/100)
                        pCoffeePot->heaterRegister = 2;
                FastForward_OneSimulationTIC(pCoffeePot);
        }
        pCoffeePot->heaterBoostRegister = 0;
        pCoffeePot->heaterRegister = 0;
        FastForward_OneSimulationTIC(pCoffeePot);
}

void Control_Both_Temp_Water ( COFFEEPOT_DEVICE* pCoffeePot, int wLevel, int hLevel){
        ShowFunctionStubInformation("Control_Both_Temp_Water");
        // water + heater
        Demonstrate_LEDControl_ASM(pCoffeePot);
        fillCoffeePotToWaterLevel_noBlock(pCoffeePot, wLevel);
        if (CurrentWaterLevel_CPP(pCoffeePot) > wLevel*4/5)      //if kinda full, start heating
                HeatWaterToTemperature_noBlock(pCoffeePot, hLevel);

        //FastForward_OneSimulationTIC(pCoffeePot);

}

double mapZeroToMax(double in, double inMax, double max) {
        double convertRatio =  inMax / max;
        if(in > inMax) in = inMax;
        return in * (max / inMax);
}

void fillCoffeePotToWaterLevel_noBlock (COFFEEPOT_DEVICE* pCoffeePot, int wLevel){
        //goes slow
        ShowFunctionStubInformation("fillCoffeePotToWaterLevel-NOBLOCK");
        //  Activate water, fill to level (blocking)
        int currentWaterLevel = CurrentWaterLevel_CPP(pCoffeePot);
        int neededWater = ( wLevel - currentWaterLevel );


        if (neededWater > 255) neededWater = 255;
        else if (neededWater < 0) neededWater = 0;

        //if (CurrentTemperature_CPP(pCoffeePot) >= 80) neededWater+=5;

        pCoffeePot->waterInFlowRegister  = neededWater;
        //printf("\n\n####### HEY PRINT THING IS %d #####\n\n",printCoffeePotInfo&1);
        if (printCoffeePotInfo && SHOW_FUNCTION_STUB_INFORMATION) printf("waterlevel %d / %d, expected
water reg %d ,water register %d\n", currentWaterLevel, wLevel, neededWater, pCoffeePot-
>waterInFlowRegister);

        if (currentWaterLevel < wLevel) {
                //if(currentWaterLevel > wLevel - 40)
                //      pCoffeePot->waterInFlowRegister = 1;
                // TODO may not need this
        } else pCoffeePot->waterInFlowRegister = 0;

        //  Activate water, fill to level (non blocking)
}

void HeatWaterToTemperature_noBlock (COFFEEPOT_DEVICE* pCoffeePot, int hLevel){
        ShowFunctionStubInformation("HeatWaterToTemperature-NOBLOCK");
        // heat to currentTemperature (blocking)
        int currentTemperature = CurrentTemperature_CPP(pCoffeePot);
        int HR,HBR,targetTempDiff = hLevel - currentTemperature + 1;

        //targetTempDiff /= (NUM_COFFEEPOTS/2 + 1);
```

```
        if (targetTempDiff < 0) targetTempDiff = 0;

        pCoffeePot->heaterBoostRegister = HBR = ((int) mapZeroToMax(targetTempDiff, hLevel, 0xd)) + 2;
        pCoffeePot->heaterRegister = HR = ((int) mapZeroToMax(targetTempDiff+35, hLevel, 0xff));

        if(printCoffeePotInfo && SHOW_FUNCTION_STUB_INFORMATION) printf("heat is %d / %d, heater is %d
/%d, heatBoost is %d / %d\n", currentTemperature, hLevel, HR, 0xff, HBR, 0xf);

        if (currentTemperature <( hLevel )) { ; ;
                //pCoffeePot->heaterRegister = 255;
                //if(currentTemperature > hLevel - 35)
                        //pCoffeePot->heaterRegister = 2;
        } else {
                pCoffeePot->heaterBoostRegister = 0;
                pCoffeePot->heaterRegister = 0;
        }
}


/*****************************************************************************
 * CoffeePot_assignment_1_Core0.h
 *****************************************************************************/

#ifndef __COFFEEPOT_ASSIGNMENT_1_CORE0_H__
#define __COFFEEPOT_ASSIGNMENT_1_CORE0_H__
        #include "../include/Sim_CoffeePot_SimulatorStructures2018.h"
        #include "../include/Sim_CoffeePot_Functions2018.h"


        //maximum of 4
        #define NUM_COFFEEPOTS 4

        static bool printCoffeePotInfo = true;

        inline void ShowCoffeePotInformation (int potNum, COFFEEPOT_DEVICE* pCoffeePot) {
                if (printCoffeePotInfo) printf("Pot %d / %d, controlRegister 0x%04x, totalWater %04d,
heaterRegister %04d, heaterBoostRegister %04d, currentTemperature %04d\n",
                                potNum, NUM_COFFEEPOTS, pCoffeePot->controlRegister,
CurrentWaterLevel_CPP(pCoffeePot),
                                pCoffeePot->heaterRegister, pCoffeePot->heaterBoostRegister,
CurrentTemperature_CPP(pCoffeePot));
        }

        void ShowNameProcessorUsed( void );

        void Init_CoffeePot(COFFEEPOT_DEVICE*);

        void Activate_LED_Control(COFFEEPOT_DEVICE*);

        void Demonstrate_LEDControl_CPP (COFFEEPOT_DEVICE*);

        void Activate_Water_Control (COFFEEPOT_DEVICE*);

        void fillCoffeePotToWaterLevel (COFFEEPOT_DEVICE*, int);

        void fillCoffeePotToWaterLevel_noBlock (COFFEEPOT_DEVICE*, int);

        void Activate_Heater_Control (COFFEEPOT_DEVICE*);

        void HeatWaterToTemperature (COFFEEPOT_DEVICE*, int);

        void HeatWaterToTemperature_noBlock (COFFEEPOT_DEVICE*, int);

        void Control_Both_Temp_Water (COFFEEPOT_DEVICE*, int, int);

#endif /* __COFFEEPOT_ASSIGNMENT_1_CORE0_H__ */
```