/\*

\*

\* 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\_\_ \*/