```c
/**
* EventCheckers.c
* This file contains the event checkers for the Mummy DDM.
*/

#include "ES_Configure.h"
#include "ES_Events.h"
#include "ES_PostList.h"
#include "ES_ServiceHeaders.h"
#include "ES_Port.h"
#include "EventCheckers.h"
#include "ES_Timers.h"
#include "ADCSWTrigger.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"

#define BEARD_INIT_VAL 1950 // initial value for beard potentiometer
#define BEARD_TOL 60 // uncertainty in beard potentiometer digital reading
#define BEARD_00 1
#define BEARD_01 488
#define BEARD_10 976
#define BEARD_11 1464
#define BEARD_20 1952
#define BEARD_21 2440

/* Module variables */
static uint8_t lastButton1State = 0; // last button 1 state
static uint8_t lastButton2State = 0; // last button 2 state
static uint8_t lastButton3State = 0; // last button 3 state
static uint8_t lastButton4State = 0; // last button 4 state
static uint8_t lastButton5State = 0; // last button 5 state
static uint8_t lastButton6State = 0; // last button 6 state
static uint16_t lastBeardState = BEARD_INIT_VAL; // last beard state
static uint8_t lastHall1AState = 1; // last Hall 1A state
static uint8_t lastHall1BState = 1; // last Hall 1B state
static uint8_t lastHall2AState = 1; // last Hall 2A state
static uint8_t lastHall2BState = 1; // last Hall 2B state
static uint8_t lastWireState = 0; // last state of first half of Udjat wire

/**
* Checks if the button 1 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton1Events(void) {
  uint8_t currentButton1State;
  bool returnVal = false;

  currentButton1State = BUTTON_1_PORT & BUTTON_1_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
  if (currentButton1State != lastButton1State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton1State == BUTTON_1_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
        }
  }
  lastButton1State = currentButton1State;
  return returnVal;
}

/**
* Checks if the button 2 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton2Events(void) {
  uint8_t currentButton2State;
  bool returnVal = false;

  currentButton2State = BUTTON_2_PORT & BUTTON_2_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
  if (currentButton2State != lastButton2State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton2State == BUTTON_2_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 2;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 2;
            PostMummySM(ThisEvent);
        }
  }
```

```
  lastButton2State = currentButton2State;
  return returnVal;
}

/**
* Checks if the button 3 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton3Events(void) {
  uint8_t currentButton3State;
  bool returnVal = false;

  currentButton3State = BUTTON_3_PORT & BUTTON_3_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
  if (currentButton3State != lastButton3State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton3State == BUTTON_3_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 3;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 3;
            PostMummySM(ThisEvent);
        }
  }
  lastButton3State = currentButton3State;
  return returnVal;
}

/**
* Checks if the button 4 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton4Events(void) {
  uint8_t currentButton4State;
  bool returnVal = false;

  currentButton4State = BUTTON_4_PORT & BUTTON_4_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
  if (currentButton4State != lastButton4State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton4State == BUTTON_4_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 4;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 4;
            PostMummySM(ThisEvent);
        }
  }
  lastButton4State = currentButton4State;
  return returnVal;
}

/**
* Checks if the button 5 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton5Events(void) {
  uint8_t currentButton5State;
  bool returnVal = false;

  currentButton5State = BUTTON_5_PORT & BUTTON_5_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
  if (currentButton5State != lastButton5State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton5State == BUTTON_5_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 5;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 5;
            PostMummySM(ThisEvent);
        }
  }
  lastButton5State = currentButton5State;
  return returnVal;
}

/**
* Checks if the button 6 has been pressed/released, based on rising/falling edges.
*/
bool CheckButton6Events(void) {
  uint8_t currentButton6State;
  bool returnVal = false;

  currentButton6State = BUTTON_6_PORT & BUTTON_6_PIN; // isolate state of pin connected to button
  /* Check for pin different from last time */
```

```
  if (currentButton6State != lastButton6State) { // event detected, so post detected event
        ES_Event ThisEvent;
        returnVal = true;
        if (currentButton6State == BUTTON_6_PIN) { // assuming that "button pressed" => high
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 6;
            PostMummySM(ThisEvent);
        } else {
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 6;
            PostMummySM(ThisEvent);
        }
  }
  lastButton6State = currentButton6State;
  return returnVal;
}

/**
* Checks if the beard has been rotated to a valid position.
*/
bool CheckBeardEvents(void) {
  uint16_t currentBeardState;
  bool returnVal = false;

  currentBeardState = ADC0_InSeq3(); // get ADC value from beard potentiometer
  if ((currentBeardState <= (lastBeardState - BEARD_TOL)) || (currentBeardState >= (lastBeardState + BEARD_TOL))) {
        ES_Event ThisEvent;
        if ((currentBeardState >= BEARD_00) && (currentBeardState <= BEARD_01)) {
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 0;
            returnVal = true;
            PostMummySM(ThisEvent);
        } else if ((currentBeardState >= BEARD_10) && (currentBeardState <= BEARD_11)) {
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 1;
            returnVal = true;
            PostMummySM(ThisEvent);
        } else if ((currentBeardState >= BEARD_20) && (currentBeardState <= BEARD_21)) {
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 2;
            returnVal = true;
            PostMummySM(ThisEvent);
        }
        lastBeardState = currentBeardState; // only update lastBeardState if potentiometer has been rotated
  }
  return returnVal;
}

/**
* Checks if events have occurred at the starting Hall effect sensors (1A and 1B).
*/
bool CheckStartHallEvents(void) {
  uint8_t currentHall1AState;
  uint8_t currentHall1BState;
  uint8_t currentWireState;
  bool returnVal = false;

  currentHall1AState = HALL_1A_PORT & HALL_1A_PIN;
  currentHall1BState = HALL_1B_PORT & HALL_1B_PIN;
  currentWireState = WIRE_PORT & WIRE_PIN;
  if ((currentHall1AState != lastHall1AState) || (currentHall1BState != lastHall1BState)) { // 1A state OR 1B state has changed
        if ((currentHall1AState == 0) && (currentHall1BState == 0) && (currentWireState == 0)) { // 1A has magnet in proximity AND 1B has
            ES_Event ThisEvent;
            ThisEvent.EventType = StartHallsFall;
            ThisEvent.EventParam = 1;
            returnVal = true;
            PostMummySM(ThisEvent);
        }
  }

  lastHall1AState = currentHall1AState;
  lastHall1BState = currentHall1BState;
  return returnVal;
}

/**
* Checks if events have occurred at the middle Hall effect sensors (2A and 2B).
*/
bool CheckMiddleHallEvents(void) {
  uint8_t currentHall2AState;
  uint8_t currentHall2BState;
  uint8_t currentWireState;
  bool returnVal = false;

  currentHall2AState = HALL_2A_PORT & HALL_2A_PIN;
  currentHall2BState = HALL_2B_PORT & HALL_2B_PIN;
  currentWireState = WIRE_PORT & WIRE_PIN;
  if ((currentHall2AState != lastHall2AState) || (currentHall2BState != lastHall2BState)) { // 2A state OR 2B state has changed
        if ((currentHall2AState == 0) && (currentHall2BState == 0) && (currentWireState == 0)) { // 2A has magnet in proximity AND 2B has
            ES_Event ThisEvent;
            ThisEvent.EventType = MiddleHallsFall;
```

```
            ThisEvent.EventParam = 1;
            returnVal = true;
            PostMummySM(ThisEvent);
        }
  }

  lastHall2AState = currentHall2AState;
  lastHall2BState = currentHall2BState;
  return returnVal;
}

/**
* Checks if loop has touched the wire in the wire loop game.
*/
bool CheckWireTouchEvents(void) {
  uint8_t currentWireState;
  bool returnVal = false;

  currentWireState = WIRE_PORT & WIRE_PIN;
  if (currentWireState != lastWireState) { // wire state has changed
        if (currentWireState != 0) { // wire high
            ES_Event ThisEvent;
            //ThisEvent.EventType = WireTouch; // no debouncing
            ThisEvent.EventType = WireTouchDB; // create debouncing event
            ThisEvent.EventParam = 1;
            returnVal = true;
            //PostMummySM(ThisEvent); // post directly
            PostWireTouchDB(ThisEvent); // post to debouce service
        }
  }

  lastWireState = currentWireState;
  return returnVal;
}

/**
* Checks for keystrokes and posts corresponding events to the Mummy state machine
*/
bool Check4Keystroke(void) {
  if (IsNewKeyReady()) { // new key waiting?
        ES_Event ThisEvent;
        ThisEvent.EventType = ES_NEW_KEY;
        ThisEvent.EventParam = GetNewKey();

        if (ThisEvent.EventParam == '1') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("1\r\n");
        } else if (ThisEvent.EventParam == '2') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 2;
            PostMummySM(ThisEvent);
            printf("2\r\n");
        } else if (ThisEvent.EventParam == '3') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 3;
            PostMummySM(ThisEvent);
            printf("3\r\n");
        } else if (ThisEvent.EventParam == '4') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 4;
            PostMummySM(ThisEvent);
            printf("4\r\n");
        } else if (ThisEvent.EventParam == '5') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 5;
            PostMummySM(ThisEvent);
            printf("5\r\n");
        } else if (ThisEvent.EventParam == '6') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonPressed;
            ThisEvent.EventParam = 6;
            PostMummySM(ThisEvent);
            printf("6\r\n");
        } else if (ThisEvent.EventParam == '!') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("!\r\n");
        } else if (ThisEvent.EventParam == '@') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 2;
            PostMummySM(ThisEvent);
```

```
            printf("@\r\n");
        } else if (ThisEvent.EventParam == '#') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 3;
            PostMummySM(ThisEvent);
            printf("#\r\n");
        } else if (ThisEvent.EventParam == '$') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 4;
            PostMummySM(ThisEvent);
            printf("$\r\n");
        } else if (ThisEvent.EventParam == '%') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 5;
            PostMummySM(ThisEvent);
            printf("%\r\n");
        } else if (ThisEvent.EventParam == '^') {
            ES_Event ThisEvent;
            ThisEvent.EventType = ButtonReleased;
            ThisEvent.EventParam = 6;
            PostMummySM(ThisEvent);
            printf("^\r\n");
        } else if (ThisEvent.EventParam == 'z') {
            ES_Event ThisEvent;
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 0;
            PostMummySM(ThisEvent);
            printf("z\r\n");
        } else if (ThisEvent.EventParam == 'x') {
            ES_Event ThisEvent;
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("x\r\n");
        } else if (ThisEvent.EventParam == 'c') {
            ES_Event ThisEvent;
            ThisEvent.EventType = BeardRotated;
            ThisEvent.EventParam = 2;
            PostMummySM(ThisEvent);
            printf("c\r\n");
        } else if (ThisEvent.EventParam == 's') {
            ES_Event ThisEvent;
            ThisEvent.EventType = StartHallsFall;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("s\r\n");
        } else if (ThisEvent.EventParam == 'm') {
            ES_Event ThisEvent;
            ThisEvent.EventType = MiddleHallsFall;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("m\r\n");
        } else if (ThisEvent.EventParam == 'w') {
            ES_Event ThisEvent;
            ThisEvent.EventType = WireTouch;
            ThisEvent.EventParam = 1;
            PostMummySM(ThisEvent);
            printf("w\r\n");
        } else if (ThisEvent.EventParam == 'a') {
            ES_Event ThisEvent;
            ThisEvent.EventType = DoorOpen;
            PostGearMotor(ThisEvent);
            printf("a\r\n");
        } else if (ThisEvent.EventParam == 'l') {
            ES_Event ThisEvent;
            ThisEvent.EventType = DoorClose;
            PostGearMotor(ThisEvent);
            printf("l\r\n");
        }
        return true;
    }
    return false;
}
```