CEC322 – 2

Lab 5: State Machine, Macro, and Assembly

Lab start date: 02/20/19

Report data: 02/20/19

Nathan Rose and Stark Cameron

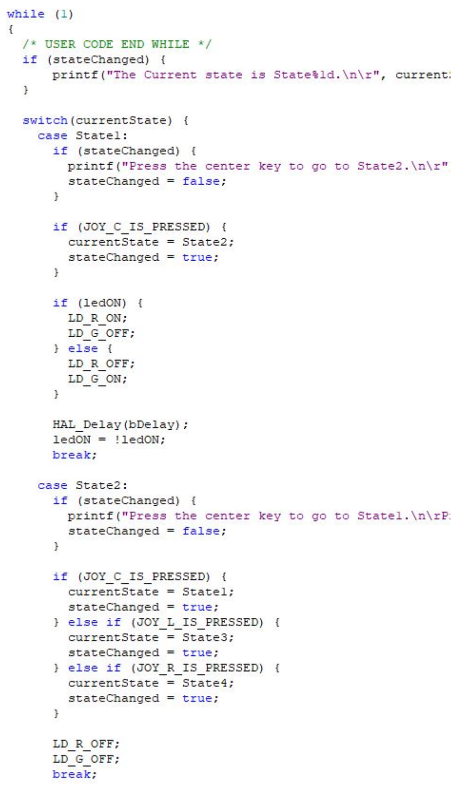
Introduction:

The goal of this assignment was to create a state machine that was controlled by the joystick and outputted different LED values and sent a string to the computer using the serial communication.

Discussions:

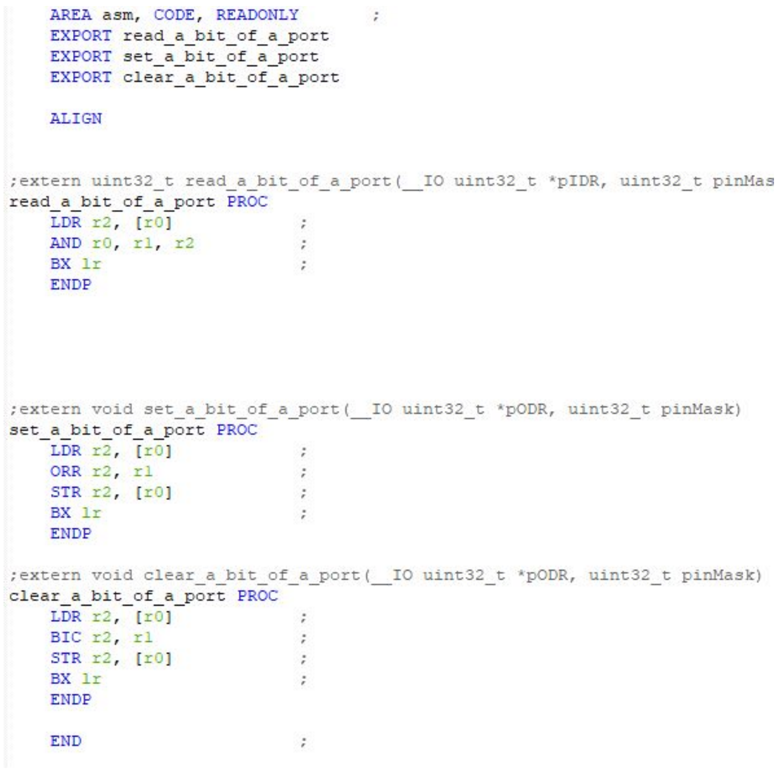
Task 1: Was to implement multiple macro functions that will later be implemented in assembly but called in the c program. These macros allow for code reusability and fixing.

Task 2: Was to create the state machine for the program, which consisted of a switch statement inside of the while loop, with flags for the current state (which corresponded with different states) and a state change flag which displayed the options for each state. Portions of state 1 and state 2 are seen below, with state 3 and 4 following the same pattern



```
while (1)
{
  /* USER CODE END WHILE */
  if (stateChanged) {
      printf("The Current state is State%ld.\n\r", current
  }

  switch(currentState) {
    case State1:
      if (stateChanged) {
        printf("Press the center key to go to State2.\n\r"
        stateChanged = false;
      }

      if (JOY_C_IS_PRESSED) {
        currentState = State2;
        stateChanged = true;
      }

      if (ledON) {
        LD_R_ON;
        LD_G_OFF;
      } else {
        LD_R_OFF;
        LD_G_ON;
      }

      HAL_Delay(bDelay);
      ledON = !ledON;
      break;

    case State2:
      if (stateChanged) {
        printf("Press the center key to go to State1.\n\rP
        stateChanged = false;
      }

      if (JOY_C_IS_PRESSED) {
        currentState = State1;
        stateChanged = true;
      } else if (JOY_L_IS_PRESSED) {
        currentState = State3;
        stateChanged = true;
      } else if (JOY_R_IS_PRESSED) {
        currentState = State4;
        stateChanged = true;
      }

      LD_R_OFF;
      LD_G_OFF;
      break;
```

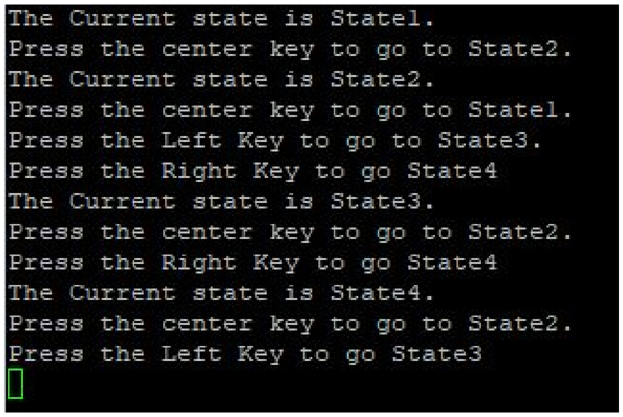
Task 3: Was to implement the assembly code which will control the state of the LEDs by setting and clearing the values and for reading the input from the joystick. The code as seen below, does the functionality.



```
    AREA asm, CODE, READONLY        ;
    EXPORT read_a_bit_of_a_port
    EXPORT set_a_bit_of_a_port
    EXPORT clear_a_bit_of_a_port

    ALIGN


;extern uint32_t read_a_bit_of_a_port(__IO uint32_t *pIDR, uint32_t pinMas
read_a_bit_of_a_port PROC
    LDR r2, [r0]                ;
    AND r0, r1, r2              ;
    BX lr                       ;
    ENDP




;extern void set_a_bit_of_a_port(__IO uint32_t *pODR, uint32_t pinMask)
set_a_bit_of_a_port PROC
    LDR r2, [r0]                ;
    ORR r2, r1                  ;
    STR r2, [r0]                ;
    BX lr                       ;
    ENDP

;extern void clear_a_bit_of_a_port(__IO uint32_t *pODR, uint32_t pinMask)
clear_a_bit_of_a_port PROC
    LDR r2, [r0]                ;
    BIC r2, r1                  ;
    STR r2, [r0]                ;
    BX lr                       ;
    ENDP

    END                         ;
```

Results:

The results of the program would be to traverse through the state via pressing of the joystick, such as while in state 1, the center key is pressed to go to state 2. In state 2, the center key for state 1, the left key for state 3 and right key for state 4. Below is the output from the controller to the putty console.



```
The Current state is State1.
Press the center key to go to State2.
The Current state is State2.
Press the center key to go to State1.
Press the Left Key to go to State3.
Press the Right Key to go State4
The Current state is State3.
Press the center key to go to State2.
Press the Right Key to go State4
The Current state is State4.
Press the center key to go to State2.
Press the Left Key to go State3
```