Text

Description automatically generated

**School of Electrical and Computer Engineering**

**Fall 2022**

**ECE 5721**

**Embedded System Design**

**Lab 3: Binary-to-Hexadecimal Seven Segment Display**

**Lab Report**

**Daniel Funke**

**November 7, 2022**

**Description:**

For this experiment I implemented a binary-to-hexadecimal converted using four input buttons, four single LED elements, and a seven-segment display. The input buttons toggled a Boolean value representing bit positions 0-3 of a four-bit binary value. Pressing a button once would set its respective bit value and pressing the button again would clear it. The value of each bit was represented by a LED, i.e if a bit was set, the corresponding LED would be turned on, and when it was cleared it would be turned off. Finally, the hexadecimal representation of the input value was shown to the user using a seven-segment display.

The switches were configured to use GPIO pins on Port D with pull-up resistors enabled. One terminal of the switch was connected to the GPIO pin while the other was tied to ground, as shown in Figure 1. Port D was configured to trigger an interrupt on the falling edge of an input signal. The interrupt handler associated with Port D was defined to determine which switch was pressed and toggle a Boolean value representing the state of the switch.

The LEDs that represented the binary input value were set up using GPIO pins on Port B. The anode of each LED was connected to the 3.3V supply rail through a 330 resistor, as shown in Figure 2. The cathode of each LED was connected to a GPIO pin on Port B. To enable an LED, its respective GPIO pin was driven low, and vice versa.

The seven-segment display device was a common cathode device, which meant that all LEDs in the device shared the same ground. This pin was tied to the ground rail of the KL25Z development board. The individual segments were connected to GPIO pins on port C, as shown in Figure 3. Illuminating individual segments required driving the respective pin high.

**Source Code:**

**main.c:**

1. #include "inputButtons.h"
2. #include "displayLED.h"
3. #include "display7Seg.h"
4. #include "MKL25Z4.h"
5. #include <stdint.h>
6. #define MASK(x) (1ul << (x))
7. void PORTD\_IRQHandler(void)
8. {
9. // If input button 1 is pressed
10. if (PORTD->ISFR & MASK(INPUT\_BUTTON\_0)) {
11. input\_buttons\_set\_state(INPUT\_BUTTON\_0);
12. }
13. // If input button 2 is pressed
14. if (PORTD->ISFR & MASK(INPUT\_BUTTON\_1)) {
15. input\_buttons\_set\_state(INPUT\_BUTTON\_1);
16. }
17. // If input button 3 is pressed
18. if (PORTD->ISFR & MASK(INPUT\_BUTTON\_2)) {
19. input\_buttons\_set\_state(INPUT\_BUTTON\_2);
20. }
21. // If input button 4 is pressed
22. if (PORTD->ISFR & MASK(INPUT\_BUTTON\_3)) {
23. input\_buttons\_set\_state(INPUT\_BUTTON\_3);
24. }
25. // Clear status flags
26. PORTD->ISFR = 0xffffffff;
27. }
28. int main(void)
29. {
30. input\_buttons\_init();
31. display\_LED\_init();
32. display\_7seg\_init();
33. \_\_enable\_irq();
34. while(1) {
35. display\_LED\_update();
36. display\_7seg\_update();
37. }
38. }

**inputButtons.c:**

1. #include "inputButtons.h"
2. #include "MKL25Z4.h"
3. #include <stdbool.h>
4. #include <stdint.h>
5. #define MASK(x) (1ul << (x))
7. enum input\_buttons {
8. INPUT\_BUTTON\_0,
9. INPUT\_BUTTON\_1,
10. INPUT\_BUTTON\_2,
11. INPUT\_BUTTON\_3,
12. NUM\_INPUT\_BUTTONS
13. };
14. static int input\_button\_port\_offset[NUM\_INPUT\_BUTTONS] = {
15. [INPUT\_BUTTON\_0] = 0,
16. [INPUT\_BUTTON\_1] = 1,
17. [INPUT\_BUTTON\_2] = 2,
18. [INPUT\_BUTTON\_3] = 3,
19. };
20. static bool input\_button\_state[NUM\_INPUT\_BUTTONS];
21. void input\_buttons\_init(void)
22. {
23. // Enable clock to Port D
24. SIM->SCGC5 |= SIM\_SCGC5\_PORTD\_MASK;
25. // Configure GPIO pins and button state
26. for (int i = 0; i < NUM\_INPUT\_BUTTONS; ++i) {
27. // Set pin function as GPIO
28. PORTD->PCR[input\_button\_port\_offset[i]] &= ~PORT\_PCR\_MUX\_MASK;
29. PORTD->PCR[input\_button\_port\_offset[i]] |= PORT\_PCR\_MUX(1);
30. // Enable pull-up resistor
31. PORTD->PCR[input\_button\_port\_offset[i]] |= PORT\_PCR\_PS\_MASK | PORT\_PCR\_PE\_MASK;
33. // Enable interrupts on rising edge
34. PORTD->PCR[input\_button\_port\_offset[i]] |= PORT\_PCR\_IRQC(0x0a);
36. // Set pin as input
37. PTD->PDDR &= ~(MASK(input\_button\_port\_offset[i]));
38. // Set initial state to false
39. input\_button\_state[i] = false;
40. }
42. // Initialize interrupt
43. NVIC\_SetPriority(PORTD\_IRQn, 2);
44. NVIC\_ClearPendingIRQ(PORTD\_IRQn);
45. NVIC\_EnableIRQ(PORTD\_IRQn);
46. }
47. uint8\_t input\_buttons\_get\_state(void)
48. {
49. uint8\_t state = 0;
50. for (int i = 0; i < NUM\_INPUT\_BUTTONS; ++i) {
51. if (input\_button\_state[i]) {
52. state |= MASK(i);
53. }
54. }
55. return state;
56. }
57. void input\_buttons\_set\_state(int button\_index)
58. {
59. input\_button\_state[button\_index] = !input\_button\_state[button\_index];
60. }

**displayLED.c:**

1. #include "displayLED.h"
2. #include"inputButtons.h"
3. #include "MKL25Z4.h"
4. #include <stdint.h>
5. #define MASK(x) (1ul << (x))
6. void display\_LED\_init(void)
7. {
8. SIM->SCGC5 |= SIM\_SCGC5\_PORTB\_MASK;
9. // Configure GPIO pins
10. for (int i = 0; i < NUM\_INPUT\_BUTTONS; ++i) {
11. // Set pin function as GPIO
12. PORTB->PCR[i] &= ~PORT\_PCR\_MUX\_MASK;
13. PORTB->PCR[i] |= PORT\_PCR\_MUX(1);
15. // Set pin as output
16. PTB->PDDR |= MASK(i);
17. // Set initial state to off
18. PTB->PSOR |= MASK(i);
19. }
20. }
21. void display\_LED\_update(void)
22. {
23. uint8\_t input\_state = input\_buttons\_get\_state();
24. for (int i = 0; i < NUM\_INPUT\_BUTTONS; ++i) {
25. if (input\_state & MASK(i)) {
26. PTB->PCOR |= MASK(i);
27. } else {
28. PTB->PSOR |= MASK(i);
29. }
30. }
31. }

**display7Seg.c:**

1. #include "display7Seg.h"
2. #include"inputButtons.h"
3. #include "MKL25Z4.h"
4. #include <stdint.h>
5. #define MASK(x) (1ul << (x))
6. enum display\_leds {
7. DISPLAY\_LED\_A,
8. DISPLAY\_LED\_B,
9. DISPLAY\_LED\_C,
10. DISPLAY\_LED\_D,
11. DISPLAY\_LED\_E,
12. DISPLAY\_LED\_F,
13. DISPLAY\_LED\_G,
14. NUM\_DISPLAY\_LEDS
15. };
16. static int display\_led\_port\_offset[NUM\_DISPLAY\_LEDS] = {
17. [DISPLAY\_LED\_A] = 11,
18. [DISPLAY\_LED\_B] = 10,
19. [DISPLAY\_LED\_C] = 6,
20. [DISPLAY\_LED\_D] = 5,
21. [DISPLAY\_LED\_E] = 4,
22. [DISPLAY\_LED\_F] = 3,
23. [DISPLAY\_LED\_G] = 0,
24. };
25. void display\_7seg\_init(void)
26. {
27. SIM->SCGC5 |= SIM\_SCGC5\_PORTC\_MASK;
28. // Configure LED GPIO pins
29. for (int i = 0; i < NUM\_DISPLAY\_LEDS; ++i) {
30. // Set pin function as GPIO
31. PORTC->PCR[display\_led\_port\_offset[i]] &= ~PORT\_PCR\_MUX\_MASK;
32. PORTC->PCR[display\_led\_port\_offset[i]] |= PORT\_PCR\_MUX(1);
33. // Set pin as output
34. PTC->PDDR |= MASK(display\_led\_port\_offset[i]);
35. // Set initial state to off
36. PTC->PCOR |= MASK(display\_led\_port\_offset[i]);
37. }
38. }
39. void display\_7seg\_update(void)
40. {
41. uint8\_t input\_state = input\_buttons\_get\_state();
42. switch(input\_state) {
43. case 0x00:
44. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
45. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
46. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]);
48. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
49. break;
50. case 0x01:
51. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]);
52. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
53. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
54. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
55. break;
56. case 0x02:
57. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
58. MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) |
59. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
61. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]);
62. break;
63. case 0x03:
64. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
65. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
66. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
68. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]);
69. break;
70. case 0x04:
71. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
72. MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
73. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
74. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]);
75. break;
76. case 0x05:
77. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
78. MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
79. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
80. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]);
81. break;
82. case 0x06:
83. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
84. MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
85. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]);
86. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]);
87. break;
88. case 0x07:
89. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
90. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]);
91. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) |
92. MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
94. break;
95. case 0x08:
96. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
97. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
98. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
99. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
100. break;
101. case 0x09:
102. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
103. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
104. MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
105. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]);
106. break;
107. case 0x0A:
108. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) |
109. MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) |
110. MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
111. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]);
112. break;
113. case 0x0B:
114. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
115. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
116. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
117. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]);
118. break;
119. case 0x0C:
120. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
121. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]);
122. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
123. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
124. break;
125. case 0x0D:
126. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
127. MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) |
128. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
129. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]);
130. break;
131. case 0x0E:
132. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]) |
133. MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) |
134. MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
136. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]);
137. break;
138. case 0x0F:
139. PTC->PSOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_A]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_E]) |
140. MASK(display\_led\_port\_offset[DISPLAY\_LED\_F]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_G]);
142. PTC->PCOR |= MASK(display\_led\_port\_offset[DISPLAY\_LED\_B]) | MASK(display\_led\_port\_offset[DISPLAY\_LED\_C]) |
143. MASK(display\_led\_port\_offset[DISPLAY\_LED\_D]);
144. break;
145. default:
146. break;
147. }
148. }

**Hardware Description:**

**Diagram, schematic

Description automatically generated**

**Figure 1: Input Switch Circuit**

**Chart, scatter chart

Description automatically generated**

**Figure 2: Binary LED display circuit**

**Diagram, schematic

Description automatically generated with medium confidence**

**Figure 3: Seven-Segment Display Circuit**

**Flow diagram:**

**Diagram

Description automatically generated**

**Figure 4: Flow diagram of the system**