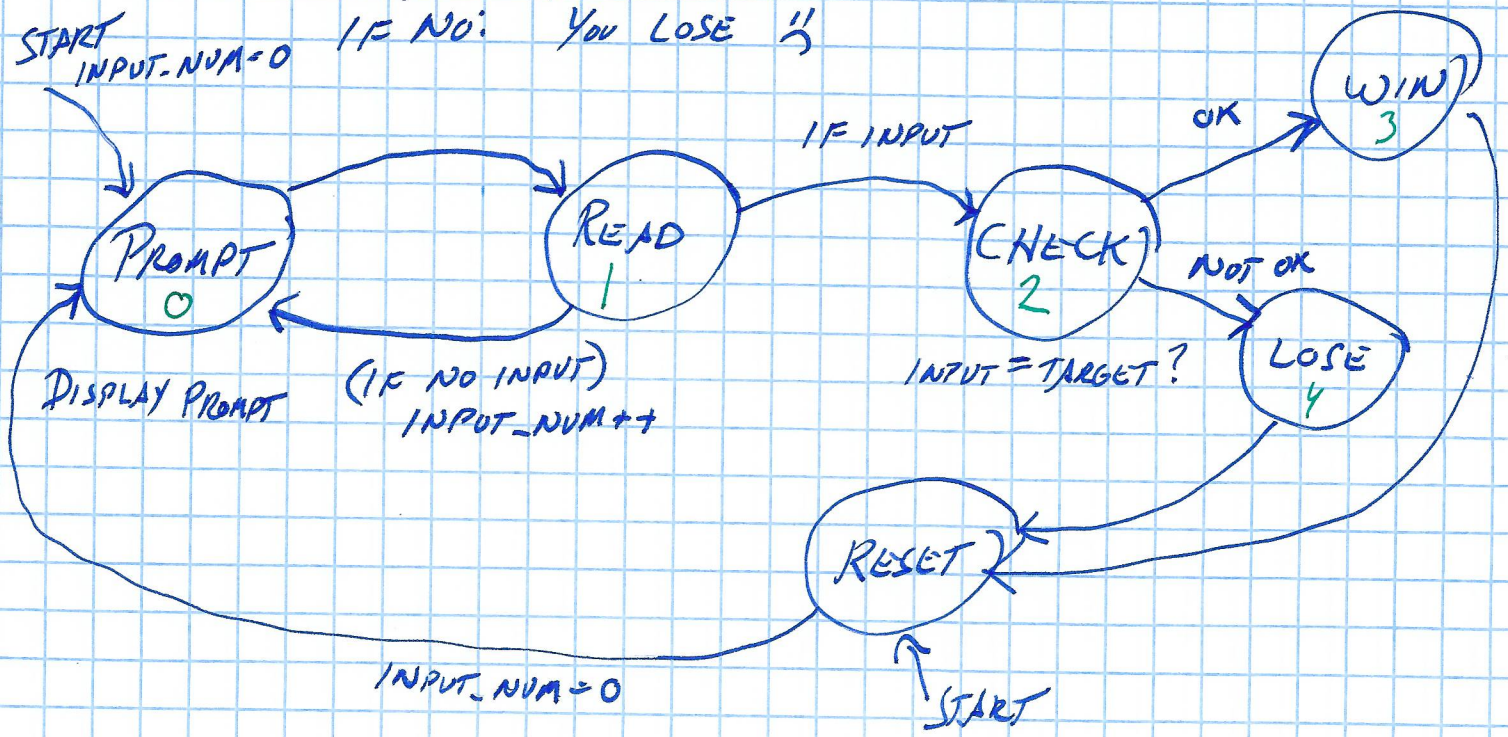
EXAMPLE: "REACTION TIME GAME"

- DISPLAY PROMPT NUMBER (IN "SCROLLING" FASHION)
- CHECK FOR INPUT ON BUTTONS
- IF INPUT, CHECK IF INPUT MATCHES TARGET
  - IF YES: YOU WIN :)
  - IF NO: YOU LOSE :(



EXAMPLE: "REACTION TIME GAME"

```c
/****** FSM EXAMPLE *******/
#include <msp430.h>

#include "peripherals.h"
#include "utils/debug_assert.h"
#include "lecture.h"

// Function prototypes
void swDelay(char numLoops);

#define BUTTON_LEFT 0x01
#define BUTTON_RIGHT 0x02

#define TARGET_NUM  7

// Here, we can represent our state as an enum, or enumerated type,
// which is a type that is functionally the same as an integer, but
// enables us to assign names to the values for clarity
enum game_state {
    PROMPT = 0,
    READ = 1,
    CHECK = 2,
    WIN  = 3,
    LOSE = 4,
    RESET = 5,
};


// Main
void main(void)
{
    enum game_state state = RESET; // Declare our game state, which starts in the RESET state
    int input_number = 0;

    WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer.

    // *** System initialization ***
    initLaunchpadButtons();
    initLaunchpadLeds();
    configDisplay();

    Graphics_clearDisplay(&g_sContext); // Clear the display
    Graphics_flushBuffer(&g_sContext);

    // *** Continued on next page ***


```

```
48
49      // To draw our prompt, we need to build an array of characters, so here
50      // we make a "buffer" of 3 characters that we will adjust based on the
    input
51      unsigned char buffer[3];
52      buffer[0] = '-'; // The first two characters make a "->" as a prompt
53      buffer[1] = '>'; // The last character will be the number
54
55      while (1)    // Forever loop
56      {
57          switch(state)
58          {
59          case PROMPT:
60              Graphics_drawStringCentered(&g_sContext, "Enter choice",
61                                          AUTO_STRING_LENGTH, 64, 15,
62                                          TRANSPARENT_TEXT);
63
64              // Add  number to the buffer, adding the offset for ASCII zero
65              buffer[2] = input_number + '0';
66              Graphics_drawStringCentered(&g_sContext, buffer, 3,
67                                          64, 25, OPAQUE_TEXT);
68              Graphics_flushBuffer(&g_sContext);
69
70              // From here, we go to the "READ" state:
71              // after the "break" statement, we re-enter the
72              // while loop and enter at the READ state
73              state = READ;
74              break;
75          case READ: {
76              char button_state = readLaunchpadButtons();
77
78              // Define next state based on input from the buttons
79              if (button_state & BUTTON_RIGHT) {
80                  state = CHECK;
81              } else {
82                  input_number = (input_number + 1) % 10;
83                  state = PROMPT;
84              }
85              break;
86          }
87          case CHECK:
88              if (input_number == TARGET_NUM) {
89                  state = WIN;
90              } else {
91                  state = LOSE;
92              }
93              break;
94
```

IF THE SIZE OF THE ARRAY IS KNOWN, WE CAN SPECIFY IT HERE. OTHERWISE, IF YOU USE AUTO_STRING_LENGTH, THE FUNCTION WILL EXPECT THE STRING TO BE NULL-TERMINATED.

WRAP NUMBER TO RANGE 0-9

```
95          case WIN:
96              Graphics_clearDisplay(&g_sContext);
97              Graphics_drawStringCentered(&g_sContext, "YAY",
98                                          AUTO_STRING_LENGTH, 64, 15,
99                                          TRANSPARENT_TEXT);
100             Graphics_flushBuffer(&g_sContext);
101             swDelay(2);
102             state = RESET;
103             break;
104         case LOSE:
105             Graphics_clearDisplay(&g_sContext);
106             Graphics_drawStringCentered(&g_sContext, "YOU FAIL",
107                                         AUTO_STRING_LENGTH, 64, 15,
108                                         TRANSPARENT_TEXT);
109             Graphics_flushBuffer(&g_sContext);
110             swDelay(2);
111             state = RESET;
112             break;
113         case RESET:
114             Graphics_clearDisplay(&g_sContext);
115             Graphics_flushBuffer(&g_sContext);
116             input_number = 0;
117
118             state = PROMPT;
119         }
120
121
122     }  // end while (1)
123 }
124
125
126
127 // ... Other demo functions omitted ...
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
```

swDelay ("Software Delay") waits for an arbitrary unit of time (not seconds!) to provide a simple delay function. We will discuss how to do accurate timing soon!