```c
#include <msp430g2211.h>

#define Reed1Mask 0b10000000
#define Reed2Mask 0b10000000

/*Since the LED is active low these are inverted in the code later, 1 means segment is on */
#define NeutralMask 0b01010100;
#define Gear1Mask 0b00000101;
#define Gear2Mask 0b01110011;
#define Gear3Mask 0b00110111;
#define Gear4Mask 0b00011101;
#define Gear5Mask 0b00111110;
#define Gear6Mask 0b01111110;
#define ErrorMask 0b01111010;

#define ALLMask 0b00100000; //Just for testing



/* very gay, but im sure Adam will love this...plus it sort of tests the LEDs at turn on */
#define A 0b01011111;
#define Line 0b00010000;
#define F 0b01011010;
#define U 0b01101101;
#define G 0b01111110;
#define E 0b01111010;

int state = 0;

int main(void) {
    /*Hold the watchdog timer */
    WDTCTL = WDTPW + WDTHOLD;

    //7 Seg setup
    P2SEL ^= Reed2Mask; //Set low to ignore the Xtal input
    P1DIR = 0x7f; //Set bits 0-6 to outputs
    P1OUT = 0x7f; //Turn all the LED's off

    //Reed Setup
    P1DIR &= ~Reed1Mask; //Insure that bit7 is set low (input)
    P2DIR &= ~Reed2Mask; //Insure that bit7 is set low (input)
    P1REN ^= Reed1Mask; //Internal Pullup
    P2REN ^= Reed2Mask; //Internal Pullup
    P1IES |= Reed1Mask; //High>Low Transition
    P2IES |= Reed2Mask; //High>Low Transition
    P1IFG &= ~Reed1Mask; //clear
    P2IFG &= ~Reed2Mask; //clear
    P1IE |= Reed1Mask; //Enable interrupts on Reed1
    P2IE |= Reed2Mask;  //Enable interrupts on Reed2

    _BIS_SR(GIE); //Enable Global Interrupts
```

```
 52
 53     //LED test
 54     P1OUT = ~A;
 55     volatile int j = 0;
 56     for(j=0; j<20000; j++);
 57     P1OUT = ~Line;
 58     for(j=0; j<20000; j++);
 59     P1OUT = ~F;
 60     for(j=0; j<20000; j++);
 61     P1OUT = ~U;
 62     for(j=0; j<20000; j++);
 63     P1OUT = ~G;
 64     for(j=0; j<20000; j++);
 65     P1OUT = ~E;
 66     for(j=0; j<20000; j++);
 67
 68
 69     while (1){
 70
 71         //state machine here
 72         switch(state) {
 73             case 0:
 74                 P1OUT = ~NeutralMask;
 75                 break;
 76             case 1:
 77                 P1OUT = ~Gear1Mask;
 78                 break;
 79             case 2:
 80                 P1OUT = ~Gear2Mask;
 81                 break;
 82             case 3:
 83                 P1OUT = ~Gear3Mask;
 84                 break;
 85             case 4:
 86                 P1OUT = ~Gear4Mask;
 87                 break;
 88             case 5:
 89                 P1OUT = ~Gear5Mask;
 90                 break;
 91             case 6:
 92                 P1OUT = ~Gear6Mask;
 93                 break;
 94             default:
 95                 P1OUT = ~ErrorMask; //Error state, hopefully won't end up
    here!
 96                 break;
 97         }
 98
 99     }
100 }
101
102 //ISRs
103 void Port_1(void) __attribute__((interrupt(PORT1_VECTOR)));
```

```
104 void Port_1(void)
105 {
106     P1IFG &= ~Reed1Mask; //Clear the interuupt
107     P1IE &= ~Reed1Mask; //Disable Interrupts to handle button bounce
108     // set the watchdog timer to trigger every 32
109     WDTCTL = WDT_MDLY_32;
110     // clear Watchdog INT flag
111     IFG1 &= ~WDTIFG;
112     // enable watchdog timer interrupts; in .... the button will be re-
    enabled
113     // by WDT_IST() --program can still run
114     IE1 |= WDTIE;
115 }
116 void Port_2(void) __attribute__((interrupt(PORT2_VECTOR)));
117 void Port_2(void)
118 {
119     P2IFG &= ~Reed2Mask; //Clear the interuupt
120     P2IE &= ~Reed2Mask; //Disable Interrupts to handle button bounce
121     // set the watchdog timer to trigger every 32ms
122     WDTCTL = WDT_MDLY_32;
123     // clear Watchdog INT flag
124     IFG1 &= ~WDTIFG;
125     // enable watchdog timer interrupts; in .... the button will be re-
    enabled
126     // by WDT_IST() --program can still run
127     IE1 |= WDTIE;
128 }
129
130 void WDT_ISR() __attribute__((interrupt(WDT_VECTOR)));
131 void WDT_ISR(void)
132 {
133     // Disable interrupts on the watchdog timer
134     IE1 &= ~WDTIE;
135     // clear the interrupt flag for the watchdog timer
136     IFG1 &= ~WDTIFG;
137     // resume holding the watchdog timer so it doesn't reset the chip
138     WDTCTL = WDTPW +WDTHOLD;
139     //after the set time check the input again and if it is still high
    then we have a debounced button
140     //if (state<6 & (!P1IN & Reed1Mask)) state++; //increments the state,
    if its not over 6, and the button is still pressed.
141     volatile int i = 0;
142     for(i=0; i<500; i++);//crappy adjustable delay, shouldn't be needed
    but the wdt delay is ruined by not having an external clock (32ms max)
143     if((P1IN & Reed1Mask) == 0)
144                         {
145                         if(state<6) state++;//increments the state, if its not
    over 6, and the button is still pressed.
146                         }
147     else if((P2IN & Reed2Mask) == 0)
148                         {
149                         if(state>0) state--;//decrements the state, if its
    over 0, and the button is still pressed.
```

```
150                     }
151     // re-enable interrupts on the button
152     P1IFG &= ~Reed1Mask; //clear any interrupts before re enable
153     P1IE |= Reed1Mask;
154     P2IFG &= ~Reed2Mask; //clear any interrupts before re enable
155     P2IE |= Reed2Mask;
156 }
157
158
159
160
```