```
#include <msp430g2211.h>

#define Reed1Mask 0b10000000;
#define Reed2Mask 0b10000000;

/*SLED is now active high in the lastest implementation, 1 means segment is on */
#define NeutralMask 0b01101101;
#define Gear1Mask 0b01000100;
#define Gear2Mask 0b01111010;
#define Gear3Mask 0b01011110;
#define Gear4Mask 0b01000111;
#define Gear5Mask 0b00011111;
#define Gear6Mask 0b00111111;
#define ErrorMask 0b00111011;

#define ALLMask 0b01111111; //used to turning off the port each time



/* very gay, but im sure Adam will love this...plus it sort of tests the LEDs at turn on */
#define A 0b01101111;
#define Line 0b00000010;
#define F 0b00101011;
#define U 0b01110101;
#define G 0b00111111;
#define E 0b00111011;

int state = 0;

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

        //7 Seg setup
        P1SEL = 0; // I/o function
        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)
        P1OUT |= Reed1Mask; //Pull High
        P2OUT |= Reed2Mask; //Pull High
        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

        //LED test
        P1OUT &= ~ALLMask; //turns off all the LEDs
        P1OUT |= A; //Puts an A on the screen by only turning on the A mask.
        volatile int j = 0;
        for(j=0; j<20000; j++);
        P1OUT &= ~ALLMask;
        P1OUT |= Line;
        for(j=0; j<20000; j++);
        P1OUT &= ~ALLMask;
        P1OUT |= F;
        for(j=0; j<20000; j++);
        P1OUT &= ~ALLMask;
        P1OUT |= U;
        for(j=0; j<20000; j++);
        P1OUT &= ~ALLMask;
        P1OUT |= G;
        for(j=0; j<20000; j++);
```

```
        P1OUT &= ~ALLMask;
        P1OUT |= E;
        for(j=0; j<20000; j++);

        while (1){

                //state machine here
                switch(state) {
                        case 0:
                                P1OUT &= ~ALLMask;
                                P1OUT |= NeutralMask;
                                break;
                        case 1:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear1Mask;
                                break;
                        case 2:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear2Mask;
                                break;
                        case 3:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear3Mask;
                                break;
                        case 4:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear4Mask;
                                break;
                        case 5:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear5Mask;
                                break;
                        case 6:
                                P1OUT &= ~ALLMask;
                                P1OUT |= Gear6Mask;
                                break;
                        default:
                                P1OUT &= ~ALLMask;
                                P1OUT |= ErrorMask; //Error state, hopefully won't end up here!
                                break;
                }
        }
}
//ISRs
void Port_1(void) __attribute__((interrupt(PORT1_VECTOR)));
void Port_1(void)
{
        P1IFG &= ~Reed1Mask; //Clear the interuupt
        P1IE &= ~Reed1Mask; //Disable Interrupts to handle button bounce
        // set the watchdog timer to trigger every 32
        WDTCTL = WDT_MDLY_32;
        // clear Watchdog INT flag
        IFG1 &= ~WDTIFG;
        // enable watchdog timer interrupts; in .... the button will be re-enabled
        // by WDT_IST() --program can still run
        IE1 |= WDTIE;
}
void Port_2(void) __attribute__((interrupt(PORT2_VECTOR)));
void Port_2(void)
{
        P2IFG &= ~Reed2Mask; //Clear the interuupt
        P2IE &= ~Reed2Mask; //Disable Interrupts to handle button bounce
        // set the watchdog timer to trigger every 32ms
        WDTCTL = WDT_MDLY_32;
        // clear Watchdog INT flag
        IFG1 &= ~WDTIFG;
        // enable watchdog timer interrupts; in .... the button will be re-enabled
        // by WDT_IST() --program can still run
        IE1 |= WDTIE;
}
void WDT_ISR() __attribute__((interrupt(WDT_VECTOR)));
void WDT_ISR(void)
```

```
{
// Disable interrupts on the watchdog timer
IE1 &= ~WDTIE;
// clear the interrupt flag for the watchdog timer
IFG1 &= ~WDTIFG;
// resume holding the watchdog timer so it doesn't reset the chip
WDTCTL = WDTPW + WDTHOLD;
//after the set time check the input again and if it is still high then we have a debounced button
//if (state<6 & (!P1IN & Reed1Mask)) state++; //increments the state, if its not over 6, and the
button is still pressed.
        volatile int i = 0;
        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)

        if((P1IN & 0b10000000) == 0){
                        if(state<6) state++; //increments the state, if its not over 6, and the button
is still pressed.
                }
        else if((P2IN & 0b10000000) == 0){
                        if(state>0) state--; //decrements the state, if its over 0, and the button is
still pressed.
                }



        // re-enable interrupts on the button
        P1IFG &= ~Reed1Mask; //clear any interrupts before re enable
        P1IE |= Reed1Mask;
        P2IFG &= ~Reed2Mask; //clear any interrupts before re enable
        P2IE |= Reed2Mask;
}
```