```c
/*
 * Initialize.h
 *
 *  Created on: Nov 11, 2016
 *      Author: aaronewing
 */

// contains all functions for Initializing MSP430

#ifndef INITIALIZE_H_
#define INITIALIZE_H_

void initialize_Joystick(void);
void initialize_Ports(void);
void initialize_Clocks(void);
void initialize_LED(void);
void initialize_Switches(void);
void initialize_Interrupts(void);

#endif /* INITIALIZE_H_ */
```

```c
/*
 * UART.h
 *
 *  Created on: Nov 10, 2016
 *      Author: aaronewing
 */

#ifndef UART_H_
#define UART_H_

void init_UART (bool baud_Rate, bool pin_Setting);               // initalizes UART clk rate and which pins are being
used
void write_UART (uint8_t TX_Data, uint8_t pin_Setting); // writes 8 bits with UART
uint8_t read_UART (void);

#endif /* UART_H_ */
```

```
/*
 * Initialize.c
 *
 *  Created on: Nov 3, 2016
 *      Author: aaronewing
 */

// contains all functions for Initializing MSP430

#include <msp430.h>
#include <stdbool.h>
#include <stdint.h>
#include "Initialize.h"


void initialize_LED(void) {
         P1DIR |= BIT0 | BIT1;                   // Sets P1.0 and P1.1 as output (LED1 and LED2)
         P1OUT &= ~(BIT0 | BIT1);    // Turns LEDs off
}


void initialize_Interrupts(void) {
//       P2IE |= BIT1 + BIT2 + BIT6;           // BIT1 = joystick left, BIT2 = joystick right, BIT6 = switch 1
          UCA0IE |= UCRXIE;                              // enable UART interrupt
         __bis_SR_register(GIE);       // enable interrupt
}


/*
 * UART.c
 *
 *  Created on: Nov 10, 2016
 *      Author: aaronewing
 */

#include <msp430.h>
#include <stdbool.h>
#include <stdint.h>
#include "UART.h"

uint8_t RX_Data = 0;

void initialize_UART(bool baud_Rate, uint8_t pin_Setting) {
         switch (pin_Setting) {
         default:
         case 0:
                   // Configure Secondary Function Pins
                   P3SEL |= BIT4 | BIT5;                                             // P3.4 - TX, P3.5 - RX

                   // assuming clk is set up already at 16MHz

                   // Configure USCI_A0 for SPI operation
                   UCA0CTL1 |= UCSWRST;                     // **Put state machine in reset**

                   switch (baud_Rate) {
                   case 0:
                            // Configure Timer for 9600 Baud
                            UCA0CTL1 = UCSSEL__ACLK;              // Set ACLK = 32768 as UCBRCLK
                            UCA0BR0 = 3;                              // 9600 baud
//                          UCA0MCTL |= 0x5300;            // 32768/9600 - INT(32768/9600)=0.41
                                                          // UCBRSx value = 0x53 (See UG)
//                          UCA0BR1 = 0;

                            UCA0BR1 = 0x00;
                            UCA0MCTL = UCBRS_3 + UCBRF_0;         // Modulation UCBRSx=3, UCBRFx=0

                            break;

                   default:
                   case 1:
                            // Configure Timer for 38400 Baud
                            UCA0CTL1 = UCSSEL__SMCLK;          // Set SMCLK = 1000000 as UCBRCLK
                            UCA0BR0 = 0x1A;                       // 9600 baud
                            UCA0MCTL |= 0x0100;      // 1000000/38400 - INT(1000000/38400)=0.04
                                                  // UCBRSx value = 0x01 (See UG)
                            // N = 0.0529, effectively 38,383.4 Baud
                            UCA0BR1 = 0;
                            break;
                   }
                   UCA0CTL1 &= ~UCSWRST; // release from reset                        // **Initialize USCI state machine**

                   break;

          case 1:
                   // Configure Secondary Function Pins
                   P5SEL |= BIT6 | BIT7;                                              // P5.6 - TX, P5.7 - RX

                   // assuming clk is set up already at 16MHz

                   // Configure USCI_A0 for SPI operation
                   UCA1CTL1 |= UCSWRST;                    // **Put state machine in reset**

                   switch (baud_Rate) {
                   case 0:
                            // Configure Timer for 9600 Baud
                            UCA1CTL1 = UCSSEL__ACLK;              // Set ACLK = 32768 as UCBRCLK
                            UCA1BR0 = 3;                              // 9600 baud
                            UCA1MCTL |= 0x5300;           // 32768/9600 - INT(32768/9600)=0.41
                                                          // UCBRSx value = 0x53 (See UG)
                            UCA1BR1 = 0;
                            break;

                   default:
                   case 1:
                            // Configure Timer for 38400 Baud
                            UCA1CTL1 = UCSSEL__SMCLK;          // Set SMCLK = 1000000 as UCBRCLK
                            UCA1BR0 = 0x1A;                       // 9600 baud
                            UCA1MCTL |= 0x0100;      // 1000000/38400 - INT(1000000/38400)=0.04
                                                                                       // UCBRSx value = 0x01 (See UG)
                            // N = 0.0529, effectively 38,383.4 Baud
                            UCA1BR1 = 0;
                            break;
                   }
                   UCA1CTL1 &= ~UCSWRST; // release from reset                        // **Initialize USCI state machine**
                   break;

          case 2:
                   // Configure Secondary Function Pins
                   P9SEL |= BIT4 | BIT5;                                              // P9.4 - TX, P9.5 - RX

                   // assuming clk is set up already at 16MHz

                   // Configure USCI_A0 for SPI operation
                   UCA2CTL1 |= UCSWRST;                    // **Put state machine in reset**

                   switch (baud_Rate) {
                   case 0:
                            // Configure Timer for 9600 Baud
                            UCA2CTL1 = UCSSEL__ACLK;              // Set ACLK = 32768 as UCBRCLK
                            UCA2BR0 = 3;                              // 9600 baud
                            UCA2MCTL |= 0x5300;           // 32768/9600 - INT(32768/9600)=0.41
                                                          // UCBRSx value = 0x53 (See UG)
                            UCA2BR1 = 0;
                            break;

                   default:
                   case 1:
                            // Configure Timer for 38400 Baud
                            UCA2CTL1 = UCSSEL__SMCLK;          // Set SMCLK = 1000000 as UCBRCLK
                            UCA2BR0 = 0x1A;                       // 9600 baud
                            UCA2MCTL |= 0x0100;      // 1000000/38400 - INT(1000000/38400)=0.04
                                                  // UCBRSx value = 0x01 (See UG)
                            // N = 0.0529, effectively 38,383.4 Baud
                            UCA2BR1 = 0;
                            break;
                   }
                   UCA2CTL1 &= ~UCSWRST; // release from reset                        // **Initialize USCI state machine**
                   break;

          case 3:
                   // Configure Secondary Function Pins
                   P10SEL |= BIT4 | BIT5;                                             // P10.4 - TX, P10.5 - RX

                   // assuming clk is set up already at 16MHz

                   // Configure USCI_A0 for SPI operation
                   UCA3CTL1 |= UCSWRST;                    // **Put state machine in reset**
```

```
                   switch (baud_Rate) {
                   case 0:
                            // Configure Timer for 9600 Baud
                            UCA3CTL1 = UCSSEL__ACLK;               // Set ACLK = 32768 as UCBRCLK
                            UCA3BR0 = 3;                               // 9600 baud
                            UCA3MCTL |= 0x5300;            // 32768/9600 - INT(32768/9600)=0.41
                                                                  // UCBRSx value = 0x53 (See UG)
                            UCA3BR1 = 0;
                            break;

                   default:
                   case 1:
                            // Configure Timer for 38400 Baud
                            UCA3CTL1 = UCSSEL__SMCLK;           // Set SMCLK = 1000000 as UCBRCLK
                            UCA3BR0 = 0x1A;                       // 9600 baud
                            UCA3MCTL |= 0x0100;      // 1000000/38400 - INT(1000000/38400)=0.04
                                                                  // UCBRSx value = 0x01 (See UG)
                            // N = 0.0529, effectively 38,383.4 Baud
                            UCA3BR1 = 0;
                            break;
                   }
                   UCA3CTL1 &= ~UCSWRST; // release from reset                       // **Initialize USCI state machine**
                   break;
         }
}

void write_UART(uint8_t TX_Data, uint8_t pin_Setting) {
         switch (pin_Setting) {
         default:
         case 0:
                   while (!(UCA0IFG & UCTXIFG)) {};                                           // If able to TX
                   UCA0TXBUF = TX_Data;                                                       // 8 bits transmitted
                   break;

         case 1:
                   while (!(UCA1IFG & UCTXIFG)) {};                                           // If able to TX
                   UCA1TXBUF = TX_Data;                                                       // 8 bits transmitted
                   break;

         case 2:
                   while (!(UCA2IFG & UCTXIFG)) {};                                           // If able to TX
                   UCA2TXBUF = TX_Data;                                                       // 8 bits transmitted
                   break;

         case 3:
                   while (!(UCA3IFG & UCTXIFG)) {};                                           // If able to TX
                   UCA3TXBUF = TX_Data;                                                       // 8 bits transmitted
                   break;
         }
}
```

```
// Aaron Ewing

#include <msp430.h>
#include <stdbool.h>
#include <stdint.h>

uint8_t RXData = 0x00;                         // global data
uint8_t TXData = 0xFF;                         // transmit data
uint8_t state = 0;
bool LED_Flag = 0;                             // LED flag

enum UART_States { RX_Data, convert_Data, TX_Data } UART_State;
void TickFct_UART() {
         switch(UART_State) {   // Transitions
         case RX_Data:
                   if (state == 0) {                                       // RX state
                            UART_State = RX_Data;
                   }
                   if (state == 1) {                                       // convert state
                            UART_State = convert_Data;
                   }
                   if (state == 2) {                                       // TX state
                            UART_State = TX_Data;
                   }
                   break;

         case convert_Data:
                   if (state == 0) {                                       // RX state
                            UART_State = RX_Data;
                   }
                   if (state == 1) {                                       // convert state
                            UART_State = convert_Data;
                   }
                   if (state == 2) {                                       // TX state
                            UART_State = TX_Data;
                   }
                   break;

         case TX_Data:
                   if (state == 0) {                                       // RX state
                            UART_State = RX_Data;
                   }
                   if (state == 1) {                                       // convert state
                            UART_State = convert_Data;
                   }
                   if (state == 2) {                                       // TX state
                            UART_State = TX_Data;
                   }
                   break;

                   default:
                            break;
         }

         switch (UART_State) {   // State actions
                   case RX_Data:
                            // do nothing
                            break;

                   case convert_Data:

                            if (RXData >= 0x61 && RXData <= 0x7A) {                 // if 'a' to 'z'
                                      TXData = RXData - 0x20;                                  // capitalize letter
                            } else {
                                      TXData = RXData;                                         // do not change input
                            }
                            state = 2;
                            break;

                   case TX_Data:
                            write_UART(TXData, 0);                                  // send TXData through UART
                            state = 0;
                            LED_Flag = 0;                                                     // turn off LED
                            break;

                   default:
                            break;
         } // State actions
}
```

```
enum LED_States { no_LED, LED } LED_State;
void TickFct_LED() {
         switch(LED_State) {   // Transitions
         case no_LED:                                // do not turn on LED
                   if (LED_Flag) {
                           LED_State = LED;
                   } else {
                            LED_State = no_LED;
                   }
                   break;

         case LED:
         if (LED_Flag) {                             // turn on LED
                  LED_State = LED;
         } else {
                  LED_State = no_LED;
         }
         default:
         break;
         }

         switch (LED_State) {        // State actions
         default:
         case no_LED:                       // turn off LED
                  P1OUT &= ~BIT0;
                  break;

         case LED:                          // turn on LED
                  P1OUT |= BIT0;
                  break;
         }
}

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;        // Stop watchdog timer

    initialize_LED();                          // initialize LEDs
    initialize_UART(0,0);            // initialize UART connection (for PC input/output)
    initialize_Interrupts();// sets up and enables UART interrupt

    while (1) {                                         // run state machine
         TickFct_UART();
         TickFct_LED();
    }
}

#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void) {
  switch(__even_in_range(UCA0IV,4)) {
  case 0:break;                             // Vector 0 - no interrupt
  case 2:                                   // Vector 2 - RXIFG
    while (!(UCA0IFG & UCTXIFG));           // USCI_A0 TX buffer ready?
    RXData = UCA0RXBUF;                        // TX -> RXData

    state = 1;                                                 // turn on LED and go to convert state
    LED_Flag = 1;
    break;
  case 4:break;                             // Vector 4 - TXIFG
  default: break;
  }
}
```