```
// Keyboard Example
// Jason Losh

//-----------------------------------------------------------------------------
// Hardware Target
//-----------------------------------------------------------------------------

// Target Platform: EK-TM4C123GXL Evaluation Board
// Target uC:       TM4C123GH6PM
// System Clock:    40 MHz

// Hardware configuration:
// Red Backlight LED:
//   PB5 drives an NPN transistor that powers the red LED
// Green Backlight LED:
//   PE4 drives an NPN transistor that powers the green LED
// Blue Backlight LED:
//   PE5 drives an NPN transistor that powers the blue LED
// Blue LED:
//   PF2 drives an NPN transistor that powers the blue LED
// Green LED:
//   PF3 drives an NPN transistor that powers the green LED
// Pushbutton:
//   SW1 pulls pin PF4 low (internal pull-up is used)
// UART Interface:
//   U0TX (PA1) and U0RX (PA0) are connected to the 2nd controller
//   The USB on the 2nd controller enumerates to an ICDI interface and a virtual COM port
//   Configured to 115,200 baud, 8N1
// 4x4 Keyboard
//   Column 0-3 outputs on PA6, PA7, PD2, PD3 are connected to cathode of diodes whose anode connects to column of keyboard
//   Rows 0-3 inputs connected to PE1, PE2, PE3, PF1 which are pulled high
//   To locate a key (r, c), the column c is driven low so the row r reads as low

//-----------------------------------------------------------------------------
// Device includes, defines, and assembler directives
//-----------------------------------------------------------------------------

#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "tm4c123gh6pm.h"
#include "wait.h"
#include "kb.h"

#define GREEN_LED    (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 3*4)))
#define BLUE_LED     (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 2*4)))
#define PUSH_BUTTON  (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 4*4)))

//-----------------------------------------------------------------------------
// Subroutines
//-----------------------------------------------------------------------------

// Blocking function that returns only when SW1 is pressed
void waitPbPress()
{
```

```
56    while(PUSH_BUTTON);
57}
58
59// Initialize Hardware
60void initHw()
61{
62    // Configure HW to work with 16 MHz XTAL, PLL enabled, system clock of 40 MHz
63    SYSCTL_RCC_R = SYSCTL_RCC_XTAL_16MHZ | SYSCTL_RCC_OSCSRC_MAIN | SYSCTL_RCC_USESYSDIV | (4
  << SYSCTL_RCC_SYSDIV_S);
64
65    // Set GPIO ports to use APB (not needed since default configuration -- for clarity)
66    // Note UART on port A must use APB
67    SYSCTL_GPIOHBCTL_R = 0;
68
69    // Enable GPIO port A, D, E, and F peripherals
70    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOA | SYSCTL_RCGC2_GPIOD | SYSCTL_RCGC2_GPIOE |
  SYSCTL_RCGC2_GPIOF;
71
72    // Configure LED and pushbutton pins
73    GPIO_PORTF_DIR_R = 0x0C;  // bits 3 and 2 are outputs, other pins are inputs
74    GPIO_PORTF_DR2R_R = 0x0C; // set drive strength to 2mA (not needed since default
  configuration -- for clarity)
75    GPIO_PORTF_DEN_R = 0x1C;  // enable LEDs and pushbuttons
76    GPIO_PORTF_PUR_R = 0x10;  // enable internal pull-up for push button
77
78    // Configure keyboard
79    // Columns 0-3 connected through diodes to PA6, PA7, PD2, PD3
80    // Rows 0-3 connected to PE1, PE2, PE3, PF1
81    GPIO_PORTA_DIR_R |= 0xC0;  // bits 6 and 7 are outputs
82    GPIO_PORTD_DIR_R |= 0x0C;  // bits 2 and 3 are outputs
83    GPIO_PORTA_DEN_R |= 0xC0;  // bits 6 and 7 are digital
84    GPIO_PORTD_DEN_R |= 0x0C;  // bits 6 and 7 are digital
85    GPIO_PORTE_DEN_R |= 0x0E;  // bits 1-3 are digital
86    GPIO_PORTF_DEN_R |= 0x02;  // bit 1 is digital
87    GPIO_PORTE_PUR_R = 0x0E;  // enable internal pull-up for rows 0-2
88    GPIO_PORTF_PUR_R = 0x02;  // enable internal pull-up for row 3
89
90    // Configure UART0 pins
91    SYSCTL_RCGCUART_R |= SYSCTL_RCGCUART_R0;         // turn-on UART0, leave other uarts in
  same status
92    GPIO_PORTA_DEN_R |= 3;                           // default, added for clarity
93    GPIO_PORTA_AFSEL_R |= 3;                         // default, added for clarity
94    GPIO_PORTA_PCTL_R = GPIO_PCTL_PA1_U0TX | GPIO_PCTL_PA0_U0RX;
95
96    // Configure UART0 to 115200 baud, 8N1 format (must be 3 clocks from clock enable and
  config writes)
97    UART0_CTL_R = 0;                                 // turn-off UART0 to allow safe
  programming
98    UART0_CC_R = UART_CC_CS_SYSCLK;                  // use system clock (40 MHz)
99    UART0_IBRD_R = 21;                               // r = 40 MHz / (Nx115.2kHz), set
  floor(r)=21, where N=16
100   UART0_FBRD_R = 45;                               // round(fract(r)*64)=45
101   UART0_LCRH_R = UART_LCRH_WLEN_8; // configure for 8N1 w/o FIFO
102   UART0_CTL_R = UART_CTL_TXE | UART_CTL_RXE | UART_CTL_UARTEN; // enable TX, RX, and module
103   UART0_IM_R = UART_IM_RXIM;                       // turn-on RX interrupt
104   NVIC_EN0_R |= 1 << (INT_UART0-16);               // turn-on interrupt 21 (UART0)
```

```
105
106     // Configure Timer 1 for keyboard service
107     SYSCTL_RCGCTIMER_R |= SYSCTL_RCGCTIMER_R1;       // turn-on timer
108     TIMER1_CTL_R &= ~TIMER_CTL_TAEN;                 // turn-off timer before reconfiguring
109     TIMER1_CFG_R = TIMER_CFG_32_BIT_TIMER;           // configure as 32-bit timer (A+B)
110     TIMER1_TAMR_R = TIMER_TAMR_TAMR_PERIOD;          // configure for periodic mode (count
    down)
111     TIMER1_TAILR_R = 0x30D40;                        // set load value to 2e5 for 200 Hz
    interrupt rate
112     TIMER1_IMR_R = TIMER_IMR_TATOIM;                 // turn-on interrupts
113     NVIC_EN0_R |= 1 << (INT_TIMER1A-16);             // turn-on interrupt 37 (TIMER1A)
114     TIMER1_CTL_R |= TIMER_CTL_TAEN;                  // turn-on timer
115 }
116
117 // Blocking function that writes a serial character when the UART buffer is not full
118 void putcUart0(char c)
119 {
120     while (UART0_FR_R & UART_FR_TXFF);
121     UART0_DR_R = c;
122 }
123
124 // Blocking function that writes a string when the UART buffer is not full
125 void putsUart0(char* str)
126 {
127     int i;
128     for (i = 0; i < strlen(str); i++)
129       putcUart0(str[i]);
130 }
131
132 // For each received character, toggle the green LED
133 // For each received "1", set the red LED
134 // For each received "0", clear the red LED
135 void Uart0Isr()
136 {
137     char c = UART0_DR_R & 0xFF;
138     if (c == '1')
139         GREEN_LED = 1;
140     if (c == '0')
141         GREEN_LED = 0;
142 }
143
144 //-----------------------------------------------------------------------------
145 // Main
146 //-----------------------------------------------------------------------------
147
148 int main(void)
149 {
150     // Initialize hardware
151     initHw();
152
153     GREEN_LED = 1;
154     waitMicrosecond(250000);
155     GREEN_LED = 0;
156     waitMicrosecond(250000);
157
158     // Display greeting
```

```
159     putsUart0("Keyboard Example\r\n");
160     putsUart0("Press '0' or '1' to turn LED on and off\r\n");
161
162     // Poll keyboard
163     char c;
164     while(1)
165     {
166         // Send characters to UART0 if available
167         if (kbhit())
168         {
169             GREEN_LED ^= 1;
170             c = getKey();
171             if (c != 'D')
172                 putcUart0(c);
173             else
174                 putsUart0("\r\n");
175         }
176         // Emulate a foreground task that is running continuously
177         BLUE_LED ^= 1;
178         waitMicrosecond(250000);
179     }
180 }
181
```