```
// UART.c
// Runs on LM3S811, LM3S1968, LM3S8962, LM4F120, TM4C123
// Simple device driver for the UART.
// Daniel Valvano
// September 11, 2013
// Modified by EE345L students Charlie Gough && Matt Hawk
// Modified by EE345M students Agustinus Darmawan && Mingjie Qiu

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2013
   Program 4.12, Section 4.9.4, Figures 4.26 and 4.40

 Copyright 2013 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// U0Rx (VCP receive) connected to PA0
// U0Tx (VCP transmit) connected to PA1

#include "UART.h"
#include "tm4c123gh6pm.h"

#define GPIO_PORTA_AFSEL_R      (*((volatile unsigned long *)0x40004420))
#define GPIO_PORTA_DEN_R        (*((volatile unsigned long *)0x4000451C))
#define GPIO_PORTA_AMSEL_R      (*((volatile unsigned long *)0x40004528))
#define GPIO_PORTA_PCTL_R       (*((volatile unsigned long *)0x4000452C))
#define UART0_DR_R              (*((volatile unsigned long *)0x4000C000))
#define UART0_FR_R              (*((volatile unsigned long *)0x4000C018))
#define UART0_IBRD_R            (*((volatile unsigned long *)0x4000C024))
#define UART0_FBRD_R            (*((volatile unsigned long *)0x4000C028))
#define UART0_LCRH_R            (*((volatile unsigned long *)0x4000C02C))
#define UART0_CTL_R             (*((volatile unsigned long *)0x4000C030))
#define UART_FR_TXFF            0x00000020  // UART Transmit FIFO Full
#define UART_FR_RXFE            0x00000010  // UART Receive FIFO Empty
#define UART_LCRH_WLEN_8        0x00000060  // 8 bit word length
#define UART_LCRH_FEN           0x00000010  // UART Enable FIFOs
#define UART_CTL_UARTEN         0x00000001  // UART Enable
#define SYSCTL_RCGC1_R          (*((volatile unsigned long *)0x400FE104))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
#define SYSCTL_RCGC1_UART0      0x00000001  // UART0 Clock Gating Control
#define SYSCTL_RCGC2_GPIOA      0x00000001  // port A Clock Gating Control

#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define UART7_DR_R              (*((volatile unsigned long *)0x40013000))
#define UART7_FR_R              (*((volatile unsigned long *)0x40013018))
#define UART7_IBRD_R            (*((volatile unsigned long *)0x40013024))
#define UART7_FBRD_R            (*((volatile unsigned long *)0x40013028))
#define UART7_LCRH_R            (*((volatile unsigned long *)0x4001302C))
#define UART7_CTL_R             (*((volatile unsigned long *)0x40013030))

#define SYSCTL_RCGC2_GPIOE      0x00000010  // port E Clock Gating Control
#define SYSCTL_RCGCUART_R7      0x00000080  // UART Module 7 Run Mode Clock

//UART1 def
#define UART1_DR_R              (*((volatile unsigned long *)0x4000D000))
#define UART1_FR_R              (*((volatile unsigned long *)0x4000D018))
#define UART1_IBRD_R            (*((volatile unsigned long *)0x4000D024))
```

```
70  #define UART1_FBRD_R            (*((volatile unsigned long *)0x4000D028))
71  #define UART1_LCRH_R            (*((volatile unsigned long *)0x4000D02C))
72  #define UART1_CTL_R             (*((volatile unsigned long *)0x4000D030))
73
74  //Port B uart1 init
75  void PortB_UART1_Init(void){
76      SYSCTL_RCGC1_R |= 0x02;               // activate UART1
77      SYSCTL_RCGCGPIO_R |= 0x02;            // activate port B
78      while((SYSCTL_PRGPIO_R&0x02) == 0){};
79      UART1_CTL_R &= ~0x01;                 // disable UART
80      UART1_IBRD_R = 86;                   // IBRD, 80Mhz clk, 38400 |
        int(80,000,000/(16*38,400)) = int(130.2083)
81      UART1_FBRD_R = 52;                    // FBRD | int(0.2083*64+0.5) = 13.8312
82
83      //UART1_IBRD_R = 52;                   // IBRD, 80Mhz clk, 19200 |
        int(80,000,000/(16*19200)) = int(260.4166)
84      //UART1_FBRD_R = 5;                    // FBRD | int(0.0833*64+0.5) = 5.3812
85      //UART1_IBRD_R = 52;                   // IBRD, 80Mhz clk, 9600 |
        int(80,000,000/(16*9600)) = int(52.0833)
86      //ART1_FBRD_R = 5;                    // FBRD | int(0.0833*64+0.5) = 5.3812
87      //UART1_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);                 // 8 bit(no
        parity, one stop, FIFOs)
88      UART1_LCRH_R = 0x0076;                 // 8 bit(no parity, one stop, FIFOs)
89
90          UART1_CTL_R |= UART_CTL_UARTEN;                  // enable UART
91      GPIO_PORTB_AFSEL_R |= 0x03;           // enable alt funct on PB0, PB1
92      GPIO_PORTB_DEN_R |= 0x03;             // enable digital I/O on PB0, PB1
93
94      GPIO_PORTB_PCTL_R &= ~0x000000FF;     // configure PB0 as U1Rx and PB1 as U1Tx
95      GPIO_PORTB_PCTL_R |= 0x44000011;
96      GPIO_PORTB_AMSEL_R &= ~0x03;          // disable analog funct on PB0, PB1
97  }
98
99  //------------UART_Init------------
100 // 8 bit word length, no parity bits, one stop bit, FIFOs enabled
101 // Input: none
102 // Output: none
103 void UART_Init(void){
104   SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART0; // activate UART0
105   SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA; // activate port A
106   UART0_CTL_R &= ~UART_CTL_UARTEN;      // disable UART
107   //UART0_IBRD_R = 43;                    // IBRD = int(80,000,000 / (16 * 115,200)) =
      int(43.40278)
108   //UART0_FBRD_R = 26;                    // FBRD = int(0.40278 * 64 + 0.5) = 26
109     UART0_IBRD_R = 86;                   // IBRD, 80Mhz clk, 38400 |
      int(80,000,000/(16*38,400)) = int(130.2083)
110     UART0_FBRD_R = 52;                    // FBRD | int(0.2083*64+0.5) = 13.8312
111                                         // 8 bit word length (no parity bits, one stop
                                            bit, FIFOs)
112   //UART0_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);
113     UART0_LCRH_R = 0x0076;
114
115   UART0_CTL_R |= UART_CTL_UARTEN;       // enable UART
116   GPIO_PORTA_AFSEL_R |= 0x03;           // enable alt funct on PA1-0
117   GPIO_PORTA_DEN_R |= 0x03;             // enable digital I/O on PA1-0
118                                         // configure PA1-0 as UART
119   GPIO_PORTA_PCTL_R = (GPIO_PORTA_PCTL_R&0xFFFFFF00)+0x00000011;
120   GPIO_PORTA_AMSEL_R &= ~0x03;          // disable analog functionality on PA
121 }
122
123 //------------UART_InChar------------
124 // Wait for new serial port input
125 // Input: none
126 // Output: ASCII code for key typed
127 unsigned char UART_InChar(void){
128   while((UART0_FR_R&UART_FR_RXFE) != 0);
129   return((unsigned char)(UART0_DR_R&0xFF));
130 }
131
```

```
unsigned char UART1_InChar(void){
  while((UART1_FR_R&UART_FR_RXFE) != 0);
  return((unsigned char)(UART1_DR_R&0xFF));
}
//------------UART_OutChar------------
// Output 8-bit to serial port
// Input: letter is an 8-bit ASCII character to be transferred
// Output: none
void UART_OutChar(unsigned char data){
  while((UART0_FR_R&UART_FR_TXFF) != 0);
  UART0_DR_R = data;
}
void UART1_OutChar(unsigned char data){
  while((UART1_FR_R&UART_FR_TXFF) != 0);
  UART1_DR_R = data;
}

//------------UART_OutString------------
// Output String (NULL termination)
// Input: pointer to a NULL-terminated string to be transferred
// Output: none
void UART_OutString(char *pt){
  while(*pt){
    UART_OutChar(*pt);
    pt++;
  }
}
void UART1_OutString(char *pt){
  while(*pt){
    UART1_OutChar(*pt);
    pt++;
  }
}
//------------UART_InUDec------------
// InUDec accepts ASCII input in unsigned decimal format
//     and converts to a 32-bit unsigned number
//     valid range is 0 to 4294967295 (2^32-1)
// Input: none
// Output: 32-bit unsigned number
// If you enter a number above 4294967295, it will return an incorrect value
// Backspace will remove last digit typed
unsigned long UART_InUDec(void){
unsigned long number=0, length=0;
char character;
  character = UART_InChar();
  while(character != CR){ // accepts until <enter> is typed
// The next line checks that the input is a digit, 0-9.
// If the character is not 0-9, it is ignored and not echoed
    if((character>='0') && (character<='9')) {
      number = 10*number+(character-'0');   // this line overflows if above 4294967295
      length++;
      UART_OutChar(character);
    }
// If the input is a backspace, then the return number is
// changed and a backspace is outputted to the screen
    else if((character==BS) && length){
      number /= 10;
      length--;
      UART_OutChar(character);
    }
    character = UART_InChar();
  }
  return number;
}

//-----------------------UART_OutUDec-----------------------
// Output a 32-bit number in unsigned decimal format
// Input: 32-bit number to be transferred
// Output: none
```

```
// Variable format 1-10 digits with no space before or after
void UART_OutUDec(unsigned long n){
// This function uses recursion to convert decimal number
//   of unspecified length as an ASCII string
  if(n >= 10){
    UART_OutUDec(n/10);
    n = n%10;
  }
  UART_OutChar(n+'0'); /* n is between 0 and 9 */
}

//---------------------UART_InUHex----------------------------------------
// Accepts ASCII input in unsigned hexadecimal (base 16) format
// Input: none
// Output: 32-bit unsigned number
// No '$' or '0x' need be entered, just the 1 to 8 hex digits
// It will convert lower case a-f to uppercase A-F
//     and converts to a 16 bit unsigned number
//     value range is 0 to FFFFFFFF
// If you enter a number above FFFFFFFF, it will return an incorrect value
// Backspace will remove last digit typed
unsigned long UART_InUHex(void){
unsigned long number=0, digit, length=0;
char character;
  character = UART_InChar();
  while(character != CR){
    digit = 0x10; // assume bad
    if((character>='0') && (character<='9')){
      digit = character-'0';
    }
    else if((character>='A') && (character<='F')){
      digit = (character-'A')+0xA;
    }
    else if((character>='a') && (character<='f')){
      digit = (character-'a')+0xA;
    }
// If the character is not 0-9 or A-F, it is ignored and not echoed
    if(digit <= 0xF){
      number = number*0x10+digit;
      length++;
      UART_OutChar(character);
    }
// Backspace outputted and return value changed if a backspace is inputted
    else if((character==BS) && length){
      number /= 0x10;
      length--;
      UART_OutChar(character);
    }
    character = UART_InChar();
  }
  return number;
}

//-------------------------UART_OutUHex----------------------------
// Output a 32-bit number in unsigned hexadecimal format
// Input: 32-bit number to be transferred
// Output: none
// Variable format 1 to 8 digits with no space before or after
void UART_OutUHex(unsigned long number){
// This function uses recursion to convert the number of
//   unspecified length as an ASCII string
  if(number >= 0x10){
    UART_OutUHex(number/0x10);
    UART_OutUHex(number%0x10);
  }
  else{
    if(number < 0xA){
      UART_OutChar(number+'0');
     }
```

```
    else{
      UART_OutChar((number-0x0A)+'A');
    }
  }
}

//------------UART_InString------------
// Accepts ASCII characters from the serial port
//    and adds them to a string until <enter> is typed
//    or until max length of the string is reached.
// It echoes each character as it is inputted.
// If a backspace is inputted, the string is modified
//    and the backspace is echoed
// terminates the string with a null character
// uses busy-waiting synchronization on RDRF
// Input: pointer to empty buffer, size of buffer
// Output: Null terminated string
// -- Modified by Agustinus Darmawan + Mingjie Qiu --
void UART_InString(char *bufPt, unsigned short max) {
int length=0;
char character;
  character = UART_InChar();
  while(character != CR){
    if(character == BS){
      if(length){
        bufPt--;
        length--;
        UART_OutChar(BS);
      }
    }
    else if(length < max){
      *bufPt = character;
      bufPt++;
      length++;
      UART_OutChar(character);
    }
    character = UART_InChar();
  }
  *bufPt = 0;
}

void UART1_InString(char *bufPt, unsigned short max) {
int length=0;
char character;
  character = UART1_InChar();
  while(character != CR){
    if(character == BS){
      if(length){
        bufPt--;
        length--;
        UART1_OutChar(BS);
      }
    }
    else if(length < max){
      *bufPt = character;
      bufPt++;
      length++;
      UART1_OutChar(character);
    }
    character = UART1_InChar();
  }
  *bufPt = 0;
}
unsigned char UART0_NonBlockingInChar(void){
    if((UART0_FR_R&UART_FR_RXFE)==0)
        return((unsigned char)(UART0_DR_R&0xFF));
    else
        return 0;
}
```

```
339
340
341   unsigned long UART1_InUDec(void){
342   unsigned long number=0, length=0;
343   char character;
344     character = UART1_InChar();
345     while(character != CR){ // accepts until <enter> is typed
346   // The next line checks that the input is a digit, 0-9.
347   // If the character is not 0-9, it is ignored and not echoed
348       if((character>='0') && (character<='9')) {
349         number = 10*number+(character-'0');   // this line overflows if above 4294967295
350         length++;
351         UART1_OutChar(character);
352       }
353   // If the input is a backspace, then the return number is
354   // changed and a backspace is outputted to the screen
355       else if((character==BS) && length){
356         number /= 10;
357         length--;
358         UART1_OutChar(character);
359       }
360       character = UART1_InChar();
361     }
362     return number;
363   }
364
365   void UART1_OutUDec(unsigned long n){
366   // This function uses recursion to convert decimal number
367   //   of unspecified length as an ASCII string
368     if(n >= 10){
369       UART1_OutUDec(n/10);
370       n = n%10;
371     }
372     UART1_OutChar(n+'0'); /* n is between 0 and 9 */
373   }
374
```