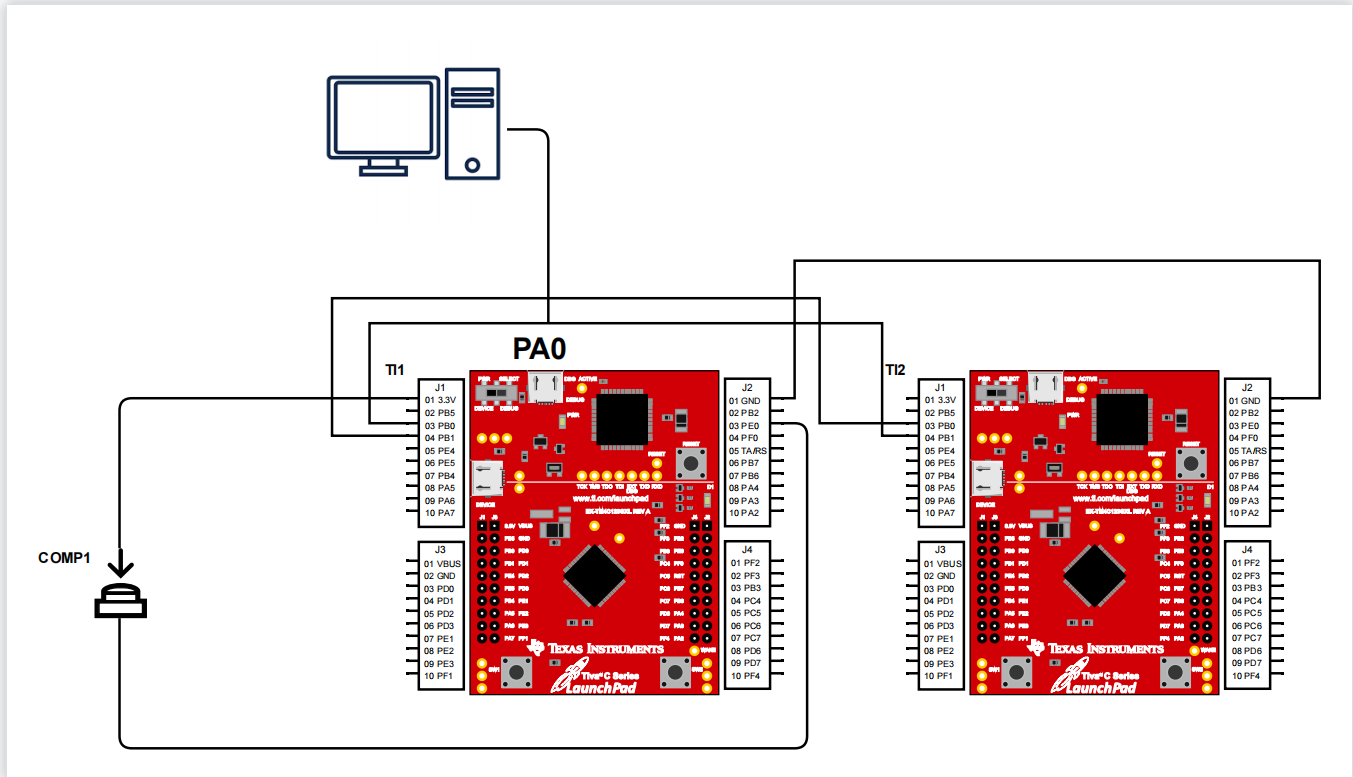
CECS 447 Project 2 UART

Introduction：In this project I wil build a communication system using UART between two TM4C123 boards and PC.

Circuit diagram：



In this project I am using four different GPIO port: A,B,E and F

Port A is used to communicate with the PC using UART0.

Port B is used to communicate between two TM4C123 MCU.

Port E is for the switch to let the user go back to the menu interface and re-select the communication mode.

Port F is for the LED and Switch on board.

Code:

UART.C

// 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

//------------UART\_Init------------

// Initialize the UART for 115,200 baud rate (assuming 50 MHz UART clock),

// 8 bit word length, no parity bits, one stop bit, FIFOs enabled

// Input: none

// Output: none

void UART\_Init(void){

SYSCTL\_RCGC1\_R |= SYSCTL\_RCGC1\_UART0; // activate UART0

SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOA; // activate port A

UART0\_CTL\_R &= ~UART\_CTL\_UARTEN; // disable UART

UART0\_IBRD\_R = 8; // IBRD = int(50,000,000 / (16 \* 115,200)) = int(27.1267)

UART0\_FBRD\_R = 44; // FBRD = int(0.1267 \* 64 + 0.5) = 8

// 8 bit word length (no parity bits, one stop bit, FIFOs)

UART0\_LCRH\_R = (UART\_LCRH\_WLEN\_8|UART\_LCRH\_FEN);

UART0\_CTL\_R |= UART\_CTL\_UARTEN; // enable UART

GPIO\_PORTA\_AFSEL\_R |= 0x03; // enable alt funct on PA1-0

GPIO\_PORTA\_DEN\_R |= 0x03; // enable digital I/O on PA1-0

// configure PA1-0 as UART

GPIO\_PORTA\_PCTL\_R = (GPIO\_PORTA\_PCTL\_R&0xFFFFFF00)+0x00000011;

GPIO\_PORTA\_AMSEL\_R &= ~0x03; // disable analog functionality on PA

}

//------------UART\_InChar------------

// Wait for new serial port input

// Input: none

// Output: ASCII code for key typed

unsigned char UART\_InChar(void){

while((UART0\_FR\_R&UART\_FR\_RXFE) != 0);

return((unsigned char)(UART0\_DR\_R&0xFF));

}

//------------UART1\_InChar------------

// Wait for new serial port input

// Input: none

// Output: ASCII code for key typed

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;

}

//------------UART1\_OutChar------------

// Output 8-bit to serial port

// Input: letter is an 8-bit ASCII character to be transferred

// Output: none

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++;

}

}

//------------UART1\_OutString------------

// Output String (NULL termination)

// Input: pointer to a NULL-terminated string to be transferred

// Output: none

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;

}

Code for transistor:

#include "UART.h"

#include "tm4c123gh6pm.h"

#define GPIO\_PORTF\_DATA\_R (\*((volatile unsigned long \*)0x400253FC))

#define GPIO\_PORTF\_DIR\_R (\*((volatile unsigned long \*)0x40025400))

#define GPIO\_PORTF\_AFSEL\_R (\*((volatile unsigned long \*)0x40025420))

#define GPIO\_PORTF\_PUR\_R (\*((volatile unsigned long \*)0x40025510))

#define GPIO\_PORTF\_DEN\_R (\*((volatile unsigned long \*)0x4002551C))

#define GPIO\_PORTF\_LOCK\_R (\*((volatile unsigned long \*)0x40025520))

#define GPIO\_PORTF\_CR\_R (\*((volatile unsigned long \*)0x40025524))

#define GPIO\_PORTF\_AMSEL\_R (\*((volatile unsigned long \*)0x40025528))

#define GPIO\_PORTF\_PCTL\_R (\*((volatile unsigned long \*)0x4002552C))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

char mode, color1,massage;

int mode1,xinhao;

int i = 0; //counter for PF0(switch 2)

int restart;

//int check = 0x01;

//int pcheck;

char lettersend;

//---------------------OutCRLF---------------------

// Output a CR,LF to UART to go to a new line

// Input: none

// Output: none

void OutCRLF(void){

UART\_OutChar(CR);

UART\_OutChar(LF);

}

//debug code

void UART1\_Init(void){

SYSCTL\_RCGCUART\_R |= 0x0002;//activate UART1

SYSCTL\_RCGCGPIO\_R |= 0x0002;//activate port B

UART1\_CTL\_R &= ~0x0001; //disable UART

UART1\_IBRD\_R = 8;//int(16\*115200)=int(27.1267)

UART1\_FBRD\_R = 44;// round(0.1267 \*64) = 8

UART1\_LCRH\_R = 0x0070;//8bit length,enable FIFO;

UART1\_CTL\_R = 0x0301;//enable RXE,TXE and UART

GPIO\_PORTB\_AFSEL\_R |= 0x03; // alt function on PB1-0

GPIO\_PORTB\_DEN\_R |= 0x03;

GPIO\_PORTB\_PCTL\_R = (GPIO\_PORTB\_PCTL\_R&0xFFFFFF00)+0x00000011;

//UART1: PB0 -> U1Rx,PB1 -> U1tx

GPIO\_PORTB\_AMSEL\_R &= ~0x03; // disable analog functionality on PB 0-1

}

void PortE\_Init(){

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x10; // 1) E clock

delay = SYSCTL\_RCGC2\_R; // delay

GPIO\_PORTE\_AMSEL\_R &= ~0x0F; // 3) disable analog function

GPIO\_PORTE\_PCTL\_R &= ~0x0000FFFF; // 4) GPIO clear bit PCTL

GPIO\_PORTE\_DIR\_R &= ~0x0F; // 5) PE3-0 input

GPIO\_PORTE\_AFSEL\_R &= ~0x0F; // 6) no alternate function

GPIO\_PORTE\_DEN\_R |= 0x0F; // 7) enable digital pins PE3-0

GPIO\_PORTE\_IS\_R &= ~0x0F;

GPIO\_PORTE\_IBE\_R &= ~0x0F;

GPIO\_PORTE\_IEV\_R &= ~0x0F;

GPIO\_PORTE\_ICR\_R = 0x0F;

GPIO\_PORTE\_IM\_R |= 0x0F;

NVIC\_PRI7\_R = (NVIC\_PRI7\_R&0xFFFF1FFF)|0x00040000; // (g)

NVIC\_EN0\_R = 0x40000000; // (h) enable interrupt 30 in NVIC

}

void PortF\_Init(void){ volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x00000020; // 1) activate clock for Port F

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

// only PF0 needs to be unlocked, other bits can't be locked

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

GPIO\_PORTF\_IS\_R &= ~0x11; // (d) PF4,PF0 is edge-sensitive

GPIO\_PORTF\_IBE\_R &= ~0x11; // PF4,PF0 is not both edges

GPIO\_PORTF\_IEV\_R &= ~0x11; // PF4,PF0 falling edge event

GPIO\_PORTF\_ICR\_R = 0x11; // (e) clear flags 4,0

GPIO\_PORTF\_IM\_R |= 0x11; // (f) arm interrupt on PF4,PF0

NVIC\_PRI7\_R = (NVIC\_PRI7\_R&0xFF1FFFFF)|0x00400000; // (g) bits:23-21 for PORTF, set priority to 2

NVIC\_EN0\_R = 0x40000000; // (h) enable interrupt 30 in NVIC

}

void colorchange(char color){

if (color == 'r')

{

GPIO\_PORTF\_DATA\_R = 0x02;

}

else if (color == 'b')

{

GPIO\_PORTF\_DATA\_R = 0x04;

}

else if (color == 'g')

{

GPIO\_PORTF\_DATA\_R = 0x08;

}

else if (color == 'y')

{

GPIO\_PORTF\_DATA\_R = 0x0A;

}

else if (color == 'w')

{

GPIO\_PORTF\_DATA\_R = 0x0E;

}

else if (color == 'p')

{

GPIO\_PORTF\_DATA\_R = 0x06;

}

else{GPIO\_PORTF\_DATA\_R = 0x00;}

}

void menu(){

UART\_OutString("Welcome to CECS 447 Project 2 - UART");

OutCRLF();

UART\_OutString("Please choose a communication mode(type 1 or 2 or 3)");

mode = UART\_InChar();

OutCRLF();

UART\_OutChar(mode);

OutCRLF();

}

void modechange(char abc){

if (abc == '1')

{

mode1 = 1;

}

else if (abc == '2')

{

mode1 = 2;

}

else if (abc == '3')

{

mode1 = 3;

}

}

void GPIOPortE\_Handler(void){

if(GPIO\_PORTE\_RIS\_R & 0x01){

GPIO\_PORTE\_ICR\_R = 0x01;

UART\_OutChar('m');

}

}

void GPIOPortF\_Handler(void){

if(GPIO\_PORTF\_RIS\_R&0x01){//PF0 is pressed

GPIO\_PORTF\_ICR\_R = 0x01;

if(i < 5){//i is a counter

i = i + 1;

}

else{

i = 0;

}

if(i == 0)

{GPIO\_PORTF\_DATA\_R = 0x00;

UART\_OutChar('d');

lettersend = 'd';

}

else if(i == 1)

{GPIO\_PORTF\_DATA\_R = 0x02;

UART\_OutChar('r');

lettersend = 'r';

}

else if(i == 2)

{GPIO\_PORTF\_DATA\_R = 0x08;

UART\_OutChar('g');

lettersend = 'g';

}

else if(i == 3)

{GPIO\_PORTF\_DATA\_R = 0x04;

UART\_OutChar('b');

lettersend = 'b';

}

else if(i == 4)

{GPIO\_PORTF\_DATA\_R = 0x06;

UART\_OutChar('p');

lettersend = 'p';

}

else if(i == 5)

{GPIO\_PORTF\_DATA\_R = 0x0E;

UART\_OutChar('w');

lettersend = 'w';

}

}

if( GPIO\_PORTF\_RIS\_R&0x10)

{

GPIO\_PORTF\_ICR\_R |= 0x10;

xinhao = 1;//triger

}

}

int main(void){

unsigned char letterget;

char string[20];

//char color[20]; // global to assist in debugging

// char color1;

PortE\_Init();

PortF\_Init();

UART\_Init(); // initialize UART

UART1\_Init();

while(1){

restart = 0;

OutCRLF();

menu();

modechange(mode);

while(mode1 == 1){

while(GPIO\_PORTE\_DATA\_R != 0x01){

UART\_OutString("enter the color: ");

color1 = UART\_InChar();

colorchange(color1);

OutCRLF();

UART\_OutString(" your color is ");

UART\_OutChar(color1);

OutCRLF();

}

break;

}

while(mode1 == 2){

while(GPIO\_PORTE\_DATA\_R != 0x01){

while(xinhao == 1){// when the PF4 is pressed,

UART1\_OutChar('m');

UART1\_OutChar(lettersend);//send the letter to Device2

UART\_OutChar(lettersend);//show the letter in the terminal

while(UART1\_InChar() != 'm');

UART\_OutString(" waitting for device 1 ");

letterget = UART1\_InChar();

UART\_OutChar(letterget);

colorchange(letterget);

//}// change the color of device 1

xinhao = 0;// reset the triger.

}

}

break;

}

while(mode1 ==3){

while(GPIO\_PORTE\_DATA\_R != 0x01){

GPIO\_PORTF\_DATA\_R = 0x08;

UART\_OutString("please enter:");

OutCRLF();

UART\_InString(string,19);

OutCRLF();

UART\_OutString(" I copy that");

UART\_OutString(string);

OutCRLF();

UART1\_OutChar('n');

UART1\_OutString(string);

}

break;

}

}

}

Code for receiver:

#include "UART.h"

#include "tm4c123gh6pm.h"

#define GPIO\_PORTF\_DATA\_R (\*((volatile unsigned long \*)0x400253FC))

#define GPIO\_PORTF\_DIR\_R (\*((volatile unsigned long \*)0x40025400))

#define GPIO\_PORTF\_AFSEL\_R (\*((volatile unsigned long \*)0x40025420))

#define GPIO\_PORTF\_PUR\_R (\*((volatile unsigned long \*)0x40025510))

#define GPIO\_PORTF\_DEN\_R (\*((volatile unsigned long \*)0x4002551C))

#define GPIO\_PORTF\_LOCK\_R (\*((volatile unsigned long \*)0x40025520))

#define GPIO\_PORTF\_CR\_R (\*((volatile unsigned long \*)0x40025524))

#define GPIO\_PORTF\_AMSEL\_R (\*((volatile unsigned long \*)0x40025528))

#define GPIO\_PORTF\_PCTL\_R (\*((volatile unsigned long \*)0x4002552C))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

char mode, color1,massage;

int mode1;

int i = 0; //counter for PF0(switch 2)

int xin= 0;

int check = 0x01;

int pcheck;

char lettersend;

//---------------------OutCRLF---------------------

// Output a CR,LF to UART to go to a new line

// Input: none

// Output: none

void OutCRLF(void){

UART\_OutChar(CR);

UART\_OutChar(LF);

}

//debug code

void UART1\_Init(void){

SYSCTL\_RCGCUART\_R |= 0x0002;//activate UART1

SYSCTL\_RCGCGPIO\_R |= 0x0002;//activate port B

UART1\_CTL\_R &= ~0x0001; //disable UART

UART1\_IBRD\_R = 8;//int(16\*115200)=int(27.1267)

UART1\_FBRD\_R = 44;// round(0.1267 \*64) = 8

UART1\_LCRH\_R = 0x0070;//8bit length,enable FIFO;

UART1\_CTL\_R = 0x0301;//enable RXE,TXE and UART

GPIO\_PORTB\_AFSEL\_R |= 0x03; // alt function on PB1-0

GPIO\_PORTB\_DEN\_R |= 0x03;

GPIO\_PORTB\_PCTL\_R = (GPIO\_PORTB\_PCTL\_R&0xFFFFFF00)+0x00000011;

//UART1: PB0 -> U1Rx,PB1 -> U1tx

GPIO\_PORTB\_AMSEL\_R &= ~0x03; // disable analog functionality on PB 0-1

}

void PortE\_Init(){

volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x10; // 1) E clock

delay = SYSCTL\_RCGC2\_R; // delay

GPIO\_PORTE\_AMSEL\_R &= ~0x01; // 3) disable analog function

GPIO\_PORTE\_PCTL\_R &= ~0x0000000F; // 4) GPIO clear bit PCTL

GPIO\_PORTE\_DIR\_R &= ~0x01; // 5) PE0 input

GPIO\_PORTE\_AFSEL\_R &= ~0x01; // 6) no alternate function

GPIO\_PORTE\_DEN\_R |= 0x01; // 7) enable digital pins PE0

}

void PortF\_Init(void){ volatile unsigned long delay;

SYSCTL\_RCGC2\_R |= 0x00000020; // 1) activate clock for Port F

delay = SYSCTL\_RCGC2\_R; // allow time for clock to start

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

// only PF0 needs to be unlocked, other bits can't be locked

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

GPIO\_PORTF\_IS\_R &= ~0x11; // (d) PF4,PF0 is edge-sensitive

GPIO\_PORTF\_IBE\_R &= ~0x11; // PF4,PF0 is not both edges

GPIO\_PORTF\_IEV\_R &= ~0x11; // PF4,PF0 falling edge event

GPIO\_PORTF\_ICR\_R = 0x11; // (e) clear flags 4,0

GPIO\_PORTF\_IM\_R |= 0x11; // (f) arm interrupt on PF4,PF0

NVIC\_PRI7\_R = (NVIC\_PRI7\_R&0xFF1FFFFF)|0x00400000; // (g) bits:23-21 for PORTF, set priority to 2

NVIC\_EN0\_R = 0x40000000; // (h) enable interrupt 30 in NVIC

}

void colorchange(char color){

if (color == 'r')

{

GPIO\_PORTF\_DATA\_R = 0x02;

}

else if (color == 'b')

{

GPIO\_PORTF\_DATA\_R = 0x04;

}

else if (color == 'g')

{

GPIO\_PORTF\_DATA\_R = 0x08;

}

else if (color == 'y')

{

GPIO\_PORTF\_DATA\_R = 0x0A;

}

else if (color == 'w')

{

GPIO\_PORTF\_DATA\_R = 0x0E;

}

else if (color == 'p')

{

GPIO\_PORTF\_DATA\_R = 0x06;

}

else{GPIO\_PORTF\_DATA\_R = 0x00;}

}

void menu(){

UART\_OutString("Welcome to CECS 447 Project 2 - UART");

OutCRLF();

UART\_OutString("Please choose a communication mode(type 1 or 2 or 3)");

mode = UART\_InChar();

OutCRLF();

UART\_OutChar(mode);

OutCRLF();

}

void modechange(char abc){

if (abc == '1')

{

mode1 = 1;

}

else if (abc == '2')

{

mode1 = 2;

}

}

char UART1\_Receiver(void){

char data;

while(UART1\_FR\_R !=0);

data = UART1\_DR\_R;

return (unsigned char) data;

}

void UART1\_TRansmitter(unsigned char data){

while(UART1\_FR\_R !=0);

UART1\_DR\_R = data;

}

void GPIOPortF\_Handler(void){

if(GPIO\_PORTF\_RIS\_R&0x01){

GPIO\_PORTF\_ICR\_R = 0x01;

if(i < 5){

i = i + 1;

}

else{

i = 0;

}

if(i == 0)

{GPIO\_PORTF\_DATA\_R = 0x00;

UART\_OutChar('d');

lettersend = 'd';

}

else if(i == 1)

{GPIO\_PORTF\_DATA\_R = 0x02;

UART\_OutChar('r');

lettersend = 'r';

}

else if(i == 2)

{GPIO\_PORTF\_DATA\_R = 0x08;

UART\_OutChar('g');

lettersend = 'g';

}

else if(i == 3)

{GPIO\_PORTF\_DATA\_R = 0x04;

UART\_OutChar('b');

lettersend = 'b';

}

else if(i == 4)

{GPIO\_PORTF\_DATA\_R = 0x06;

UART\_OutChar('p');

lettersend = 'p';

}

else if(i == 5)

{GPIO\_PORTF\_DATA\_R = 0x0E;

UART\_OutChar('w');

lettersend = 'w';

}

}

if( GPIO\_PORTF\_RIS\_R&0x10)

{

GPIO\_PORTF\_ICR\_R |= 0x10;

//GPIO\_PORTF\_DATA\_R = 0x0E;

xin = 100;

}

}

int main(void){

unsigned char i,letterget;

char string[20];

PortF\_Init();

UART\_Init(); // initialize UART

UART1\_Init();

while(1){

while(UART1\_InChar() == 'm'){

letterget = UART1\_InChar();

//UART\_OutChar(letterget);

colorchange(letterget);

while(xin != 100);

UART1\_OutChar('m');

UART\_OutChar('m');

UART1\_OutChar(lettersend);

UART\_OutChar(lettersend);

xin = 0;

}

while(UART1\_InChar() == 'n'){

letterget = UART1\_InChar();

UART\_OutChar(letterget);

}

}

//}

}